

AD-A108 253

RAYTHEON CO WAYLAND MA EQUIPMENT DIV

F/G 17/9

R&D EQUIPMENT INFORMATION REPORT. FAULT TOLERANT WEATHER RADAR --ETC(11)

MAR 81 M J YOUNG, A J JABOONIK

F1962A-7R-C-0113

UNCLASSIFIED

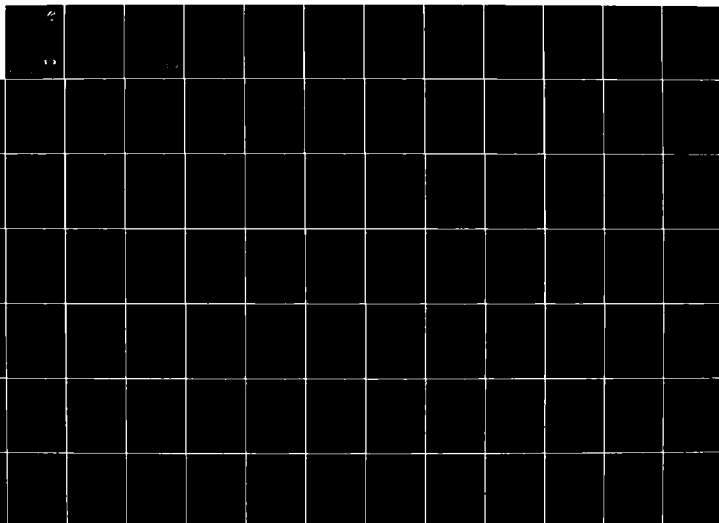
ER81-4053

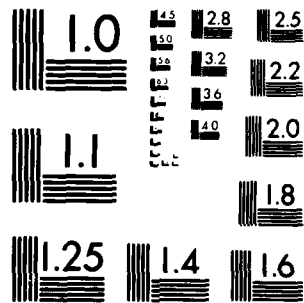
AFGL-TR-81-0086

NL

1 4

AD-A108 253





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A.

AD A108253

AFGL-TR-81-0088

②
LEVEL

R&D EQUIPMENT INFORMATION REPORT
Fault Tolerant Weather Radar Processor

Michael J. Young
Anthony S. Jagodnik, Jr.

~~Raytheon Company~~ *Wise*
Equipment Division
Advanced Systems Laboratory
Wayland, MA 01778

March 1981

Final Report
78 June 1 - 80 October 16

Approved for public release; distribution unlimited.

FILE COPY

U.S. AIR FORCE RESEARCH LABORATORY
WRIGHT-PATTERSON AIR FORCE BASE
DAYTON, OHIO 45433-6151

DTIC
ELECTRIC

15 OF 100

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFGL-TR-81-0086	2. GOVT ACCESSION NO. AD-110825	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) R & D Equipment Information Report Fault Tolerant Weather Radar Processor		5. TYPE OF REPORT & PERIOD COVERED Final Report 78 June 1 - 80 October 16
7. AUTHOR(s) Michael J. Young Anthony J. Jagodnik, Jr.		6. PERFORMING ORG. REPORT NUMBER ER81-4053
9. PERFORMING ORGANIZATION NAME AND ADDRESS Raytheon Company Equipment Division Advanced Systems Laboratory Wayland, MA 01778		8. CONTRACT OR GRANT NUMBER(s) F19628-78-C-0113
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Geophysics Laboratory Hanscom AFB, Massachusetts 01731 Monitor/Kenneth J. Banis (LYW)		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS F78100 6670064A
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE 81 March
		13. NUMBER OF PAGES 346
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE NA
16. DISTRIBUTION STATEMENT (of this Report) Approved for Public Release; Distribution Unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Weather Radar Signal Processing, Digital Signal Processing, Programmable Signal Processing, Distributed Processing, Fault Tolerant Processing		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) A programmable signal processor which achieves fault tolerance and modu- larity through distribution of tasks among a number of identical programmable Common Elements including spares, is described. This processor's design and usage in a meteorological Doppler radar signal processing application is discussed at several levels of detail covering hardware firmware, operating system soft- ware, user software, and development aids.		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

388201

1

ACKNOWLEDGEMENTS

The Fault-Tolerant Weather Radar Processor represents the combined efforts of many people from a number of laboratories within Raytheon Equipment Division. Although it is by no means exhaustive, the following list identifies those who contributed significantly to the task:

Advanced Systems Laboratory

E. M. Avery
R. J. Bonneau
J. W. Burgarella
H.E.T. Connell
L. L. DiMaria
F. P. Eggleton
R. C. Evett
V. E. Follansbee
A. J. Jagodnik
V. A. Jelich
N. F. Lacey
J. C. Murray
H. R. Riggert
D. A. Syiek
G. A. Works
M. J. Young

Radar Systems Laboratory

R. H. Cantwell
J. DelRio
M. A. Jones
A. J. Scungio
R. R. Smith

Accession For	
DTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

DTIC
ELECTE
S DEC 9 1981 D
D

TABLE OF CONTENTS

<u>Section</u>	<u>Page</u>
1. BACKGROUND	9
1.1 Meteorological Doppler Radar	9
1.2 The Pulse Pair Processor	10
1.3 The Fault-Tolerant Weather Radar Processor	10
2. SYSTEM OVERVIEW	13
2.1 Achieving Reliable Systems	13
2.2 Features of FTSP	15
2.3 Fault Tolerance Philosophy and Implementation	18
2.4 FTWRP Hardware	19
2.5 FTWRP Software	21
2.5.1 Operating System Software	21
2.5.2 FTWRP Command Processor	21
2.5.3 Signal Processing Application Programs	22
3. FTWRP SYSTEM USAGE	28
3.1 FTWRP System Startup	28
3.2 IDOS-1 FTWRP Command Interpreter	29
3.2.1 FTWRP Parameters	30
4. DETAILED HARDWARE DESCRIPTION	40
4.1 Common Element	40
4.1.1 General Description	40
4.1.2 Specification - Common Element Mark 100A	42
4.1.3 Detailed Hardware Description	44
4.2 Input/Output Controller (IOC)	58
4.3 Terminal Interface Element	64
4.3.1 Outputs from TIE	66
4.3.2 Inputs to TIE	66
4.3.3 Transceiver Status Word	68
4.4 Input Synchronizer	71
4.4.1 Input Synchronizer Hardware Description	76
4.4.2 Data Transfer Timing	78
4.5 Output Synchronizer	80
4.5.1 Output Synchronizer Hardware Description	80
4.5.2 Data Transfer Timing	84

TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Page</u>
5. DETAILED SOFTWARE DESCRIPTION	88
5.1 Distributed Operating System Level 0	88
5.1.1 Initialization	88
5.1.2 Task Environment	90
5.1.3 Input/Output Structure	94
5.1.4 System Call Processing	101
5.1.5 Trace Handling	108
5.1.6 Clock Handling	109
5.1.7 Exception Handling	109
5.1.8 Common Element Link Stack	110
5.2 Intecolor Resident DOS-0	111
5.2.1 IDOS-0 Operation	113
5.2.2 Input/Output Operation	118
5.2.3 System Services	134
5.3 Distributed Operating System Level 1	137
5.3.1 System Configuration	137
5.3.2 FTSP Level 1 Operating System Functions	137
5.4 Pulse-Pair Application Program	144
5.4.1 Overview of FTWRP Processing	144
5.4.2 FTWRP Functional Description	145
5.5 Dual Wavelength Application Task	155
5.5.1 Range Ambiguity Resolver	155
5.5.2 Coherent Channel Formatter	157
5.6 Spare Rotation	159
5.6.1 FTWRP Spare Rotation Environment	159
5.6.2 Spare Rotation Implementation	160
6. FIRMWARE DESCRIPTION	163
6.1 New Common Element Microcode	163
6.1.1 Vector Add	163
6.1.2 Vector Subtract	164
6.1.3 Vector Multiply	164
6.1.4 Block Integration	164
6.1.5 Sliding Window Integration	165
6.1.6 Scale	166
6.1.7 Circular Vectoring	166
6.1.8 Read and Accumulate	167
6.1.9 Read and Autocorrelate Initial Results	169

TABLE OF CONTENTS (Continued)

<u>Section</u>		<u>Page</u>
6.1.10	Read and Autocorrelate	171
6.2	IOC Firmware Development	171
6.2.1	Data Flow Overview	172
6.2.2	Exception Handling	172
6.2.3	IOC Continuous Input Mode Setup	174
6.2.4	Continuous Input Mode Functional Description	175
APPENDICES		
A	Intecolor/CE Support	177
B	Subroutines and Data Structures in FTWRP	253
C	"Notes on Circular Vectoring"	304
D	FTWRP System Timing Considerations	310
E	Look-Up Tables	314
F	Cables and Interconnect Specifications	334
G	FTWRP Hardware and Software Configuration	344

LIST OF ILLUSTRATIONS

<u>Figure</u>		<u>Page</u>
2-1	Efficacy of Redundant Configurations	14
2-2	FTSP Distributed Operating System	17
2-3	FTWRP Hardware Block Diagram	20
2-4	Processing Flow Diagram for Continuous Pulse Sequence	23
2-5	Processing Flow Diagram for Dual Wavelength Mode	25
4-1	Simplified Common Element Block Diagram	41
4-2	Bus Transceiver Block Diagram	45
4-3	Transmitter Timing	46
4-4	Receiver Timing	47
4-5	Common Element Detailed Block Diagram	54
4-6	Texas Instruments 74S481 Functional Block Diagram	55
4-7	Common Element Two-Phase Clock	59
4-8	IOC Module Block Diagram	60
4-9	IOC Controller Block Diagram	63
4-10	Terminal Interface Element Block Diagram	65
4-11	Distribution of Processing in FTWRP	72
4-12	Input Synchronizer Block Format	73
4-13	Continuous Pulse Scheme Timing	74
4-14	Input Synchronizer Block Diagram	77
4-15	Input Synchronizer Timing	79
4-16	Format of CE Output Buffer for NRC	81
4-17	Output Synchronizer Block Diagram	82
4-18	Output Synchronizer Input Sequencer and Buffer Write Timing	85
5-1	DOS-0 Hierarchy	89
5-2	Common Element Memory Utilization	91
5-3	FTSP Message Packet Format	95
5-4	Message Header Format	94
5-5	Text Boundary Protection	117
5-6	IDOS-0 Text Message Format	124
5-7	Valid Text Characters	125
5-8	Executive Message Format	127
5-9	Page Map Format	127
5-10	Look-Up Table Load Map Page Format	147
5-11	Range Ambiguity Resolver Flow Diagram	156
5-12	Coherent Channel Formatter Flow Diagram	158
6-1	RACOR and RACORI Data Format	170
6-2	Configuration Control Message for Continuous Input Mode	173

LIST OF TABLES

<u>Table</u>		<u>Page</u>
3-1	FTWRP Command Description	31
3-2	'SET' Parameter Mnemonics	36
3-3	Scale Constant Default Determination	38
3-4	System Limitations on NRC, TP, and NSI	39
4-1	Transmitter State	51
4-2	Last Transmit Bus	51
4-3	Receiver State	51
4-4	Block Length	51
4-5	Receiver State	69
4-6	Last Transmit Bus	69
4-7	Transmitter State	69
4-8	NRC and Tp Definitions	75
5-1	CE Task Prologue	92
5-2	Message Code Definitions	96
5-3	Message Formats	97
5-4	System Requests - DOS-0	98
5-5	FTWRP Logical Device Numbers	106
5-6	Intecolor Memory Usage	112
5-7	Intecolor User Task Prologue	114
5-8	IDOS-0 Input Buffer Packet Structure	119
5-9	IDOS-0 Message Code Processing	121
5-10	Executive Message Types	128
5-11	IDOS-0 Service Calls	130
5-12	Intecolor Input/Output Error Messages	132
5-13	Signal Processor Command Formats	149
6-1	CVEC Accuracy	168

1. BACKGROUND

1.1 Meteorological Doppler Radar

Radar has been in operational use for many years to estimate atmospheric water content over areas of tens of thousands of square miles. Such measurements are typically achieved using noncoherent, pulsed, mechanically-scanned, pencil-beam radars capable of measuring the logarithmic amplitude of precipitation echoes, followed by signal processors which average the returns over many pulses in order to enhance signal to noise ratios. These real-time signal processors typically fill their displays with about 10^5 picture elements of data derived from perhaps 10^8 individual measurements on each rotation of the antenna. But, despite the large amount of data, these systems lack the needed capability for directly measuring the velocity of their targets.

In recent years, extensive experimental work to assess the effectiveness of Doppler techniques which add this new dimension to the wide-area measurement capabilities of meteorological radars has been undertaken in a number of organizations, including the Air Force Geophysics Laboratory (AFGL). These experiments have shown that Doppler capability is indeed worthwhile and can, by adding velocity information to the meteorologist's repertoire of inputs, significantly improve the lead time and correctness of tornado and other local severe-weather advisories. But the need to make Doppler measurements places more stringent requirements on the radar (where narrower antenna beams, coherency, and linearity are required to permit phase measurement) and on the signal processor, which must perform more difficult calculations at high throughput.

Another difficulty in Doppler systems is the unambiguous range-velocity product $c\lambda/8$ for uniform-PRF systems where λ is the radar wavelength. Since λ is usually fixed by antenna-size and propagation considerations (10-cm is generally considered minimum), the actual range-velocity product falls short of what is needed by about a factor of four. Thus, the PRF can be chosen to

cover either the desired range or velocity, but not both. Means of extending the useful measurement capabilities by resolving these ambiguities is an active area of research where programmable signal processors are needed.

1.2 The Pulse Pair Processor

The input to a meteorological Doppler signal processor is a series of complex samples in the time domain, but it is the mean velocity of scatterers (average Doppler frequency shift of the returns from the sensitivity volume) which is of primary interest, so the first job of the processor is to estimate that mean velocity. One algorithm consists of transforming the data record for each range cell to the frequency domain (computing its spectrum), then performing the moment calculations. But even the computationally-efficient FFT requires $O(N \log N)$ complex multiply/additions for an N -point transform. An algorithm which operates entirely in the time-domain to estimate the mean velocity was introduced by Rummler (Reference 1-1) and has become widely used for meteorological signal processing. This pulse-pair algorithm is not only computationally more efficient (needing only $O(N)$ complex multiply/adds), but also provides better performance at low signal-to-noise ratios, as shown by Berger and Groginsky (Reference 1-2) and Sirmans and Bumgarner (Reference 1-3). A special-purpose hardware implementation of a Pulse Pair Processor (PPP) was constructed for AFGL in 1973 under contract F19628-72-C-0293. This PPP, as described by Novick and Glover (Reference 1-4), estimates mean Doppler velocity in each of up to 1024 range cells.

1.3 The Fault-Tolerant Weather Radar Processor

While the original PPP is still operational and has performed well, more flexibility was needed in order to verify new algorithms, especially those related to ambiguity resolution. It was at the same time necessary to demonstrate that this type of signal processing problem can be partitioned for implementation in a distributed processor for reasons of maintainability and fault tolerance in planned future operational systems.

* $O(N)$ is defined as "on the order of N ".

Meanwhile, Raytheon IR&D programs to develop Fault Tolerant Signal Processing (FTSP) systems using programmable Common Elements and related hardware and software were underway. The application of these developments to the weather radar processing problem promised to solve both the flexibility and maintainability problems. Thus, development of the Fault Tolerant Weather Radar Processor (FTWRP) was undertaken to implement the pulse pair and other algorithms in the FTSP hardware.

REFERENCES

- 1-1. Rummler, W.D., "Two-Pulse Spectral Measurement", technical memo MM-68-4121-15, Bell Telephone Laboratories, Whippany, NJ, 7 November 1968.
- 1-2. Berger, T. and Groginsky, H.L., "Estimates of Spectral Moments of Pulse Trains", presented at the International Conference on Information Theory, Tel-Aviv, Israel, 1973.
- 1-3. Sirmans, D. and Bumgarner, W.C., "Numerical Comparison of Five Mean Frequency Estimators", Journal of Applied Meteorology, Vol. 14, No. 6, September 1975, pp 991-1003.
- 1-4. Novick, L.R. and Glover, K.M., "Spectral Mean and Variance Estimation via Pulse Pair Processing", preprint volume, 16th Radar Meteorology Conference, April 22-24, 1975, Houston, TX.

2. SYSTEM OVERVIEW

With the goal of providing several levels of detail so that readers having differing objectives in the use of this document can seek their own level, the structure has been set up as follows. This section contains an overview of the system architecture with enough detail and philosophy background to understand FTWRP for the purposes of operating the system and understanding its relationship with existing equipment. Section 3 is intended to provide step-by-step operating instructions. Greater detail on Hardware, Software, and Firmware appears in Sections 4, 5 and 6 which reference other information contained in the Appendix.

2.1 Achieving Reliable Systems

Methods of achieving reliable systems include two major approaches: 1) use of reliable components in the system, and 2) design of redundant components into the system. The latter approach, given an effective means of locating failed elements and replacing them from a supply of spares, allows highly reliable systems to be built from ordinary components. As Figure 2-1 shows, highest probability of survival results when the system is partitioned into a large number of identical, simple modules, each of which can perform the function of any other (Reference 2-1). For maximum effectiveness, this type of redundant system should meet the following requirements:

1. The number of module types and complexity of modules should be minimized.
2. Each module must have internal fault-detection capability.
3. A means of replacing failed modules must exist.
4. The task must be partitionable without excessive inter-module communication bandwidth requirements.

Requirements 1 and 2 are best met with programmable processors, which also improve adaptability to mission changes.

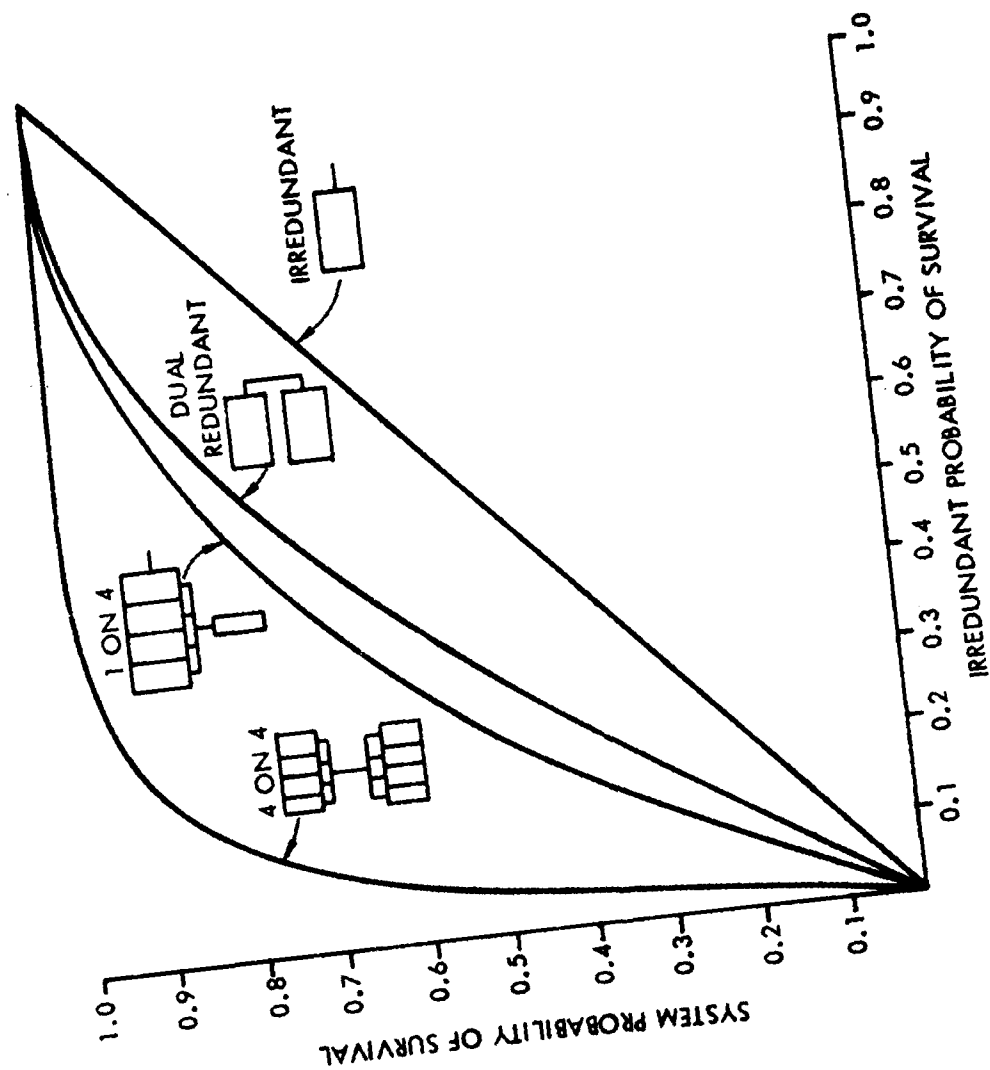


Figure 2-1. Efficacy of Redundant Configurations

The Fault-Tolerant Signal Processor (FTSP) is an example of a self-repairing programmable signal processor and its application to a meteorological Doppler radar, in which fault-tolerance, programmability, and expandability are all important features. FTSP utilizes the basic concept of distributed processing with distributed control, with the intent that total decentralization minimizes the occurrences and effects of single-point failures.

2.2 Features of FTSP

This subsection gives an overview of the Fault-Tolerant Signal Processor; more detailed descriptions of hardware, software, and firmware appear in Sections 4, 5 and 6.

The FTSP utilizes a fully-distributed architecture with three types of elements or modules: the Common Element (CE), the Common Memory (CM), and the Input/Output Controller (IOC). These modules are interconnected by dual-redundant, high-bandwidth (up to 5×10^6 16-bit words/sec) open collector busses. Dual-redundant power busses provide fault-tolerant power to each module.

The CE is a 16-bit, microprogrammed computer-on-a-card with an internal architecture and instruction set which were optimized for signal processing. All computational and network management software resides in the CE. Although the CE has 16K words of on-board memory, in many applications a larger, common data base is required. The CM fills this need by supplying 128K words of random-access memory, addressable by 240-word pages. The CM has its own intelligent microprogrammed controller for more efficient operation.

The IOC may serve either of two functions: a general-purpose parallel I/O channel to the outside world, or a "bus extender" to connect two or more clusters of elements. When used as an I/O channel, the IOC is capable of interfacing up to 16 external devices on each of two parallel busses. When used as a bus extender the IOC forwards messages from one bus to another, thus acting as a communications link between clusters of elements.

Each element communicates on a bus via message packets of up to 256 words. The first words of each packet are a header block containing information regarding the source element (sender), destination element (receiver), any required waypoint elements (bus extender IOCs), and the type of message. If the communication is within a single cluster, only one header word is required. Otherwise, the first eight words are reserved as header.

Each element has a 6-bit "virtual address" (VA) by which it is accessed on the busses. At power-up, or after a reset operation, the module receives a position-dependent "socket address." When the module is assigned a specific task, it is given a new address appropriate to that function. Therefore, if a spare must take over the task of a failed element, the executive merely resets the faulty module and changes the spare's address to that of the faulty one. The system looks the same to the rest of the elements; therefore configuration information need not change. This reconfiguration method minimizes bus traffic and overhead.

A special mechanism is provided to override control of an element, if necessary. The bus interface on each element includes a decoder to recognize certain special commands from the executive operating system. These commands can automatically reset a module to its initial power-up state, or even turn off power to an element to remove pathological faults from the system. These commands are decoded with minimal hardware, and do not require the cooperation of the element's controller or software.

The operating system of the FTSP is distributed throughout the CEs of the processor. The Distributed Operating System (DOS) is a two-level hierarchy (Figure 2-2). DOS-0, which resides in every CE, maintains local control over operations such as I/O and user services. The system executive resides in only two CEs, and actually runs as a task under the aegis of DOS-0. The active executive, (DOS-1) resides in one CE under a dedicated virtual address. An alternate executive (ADOS) resides in another CE, and acts as a watchdog over DOS-1; if DOS-1 fails, ADOS takes over and assigns its own task to another CE. DOS-1 handles global control functions, such as system configuration, partitioning of tasks, and, in the case of faults, replacement of failed elements. In FTWRP, DOS-1 resides in an intelligent terminal.

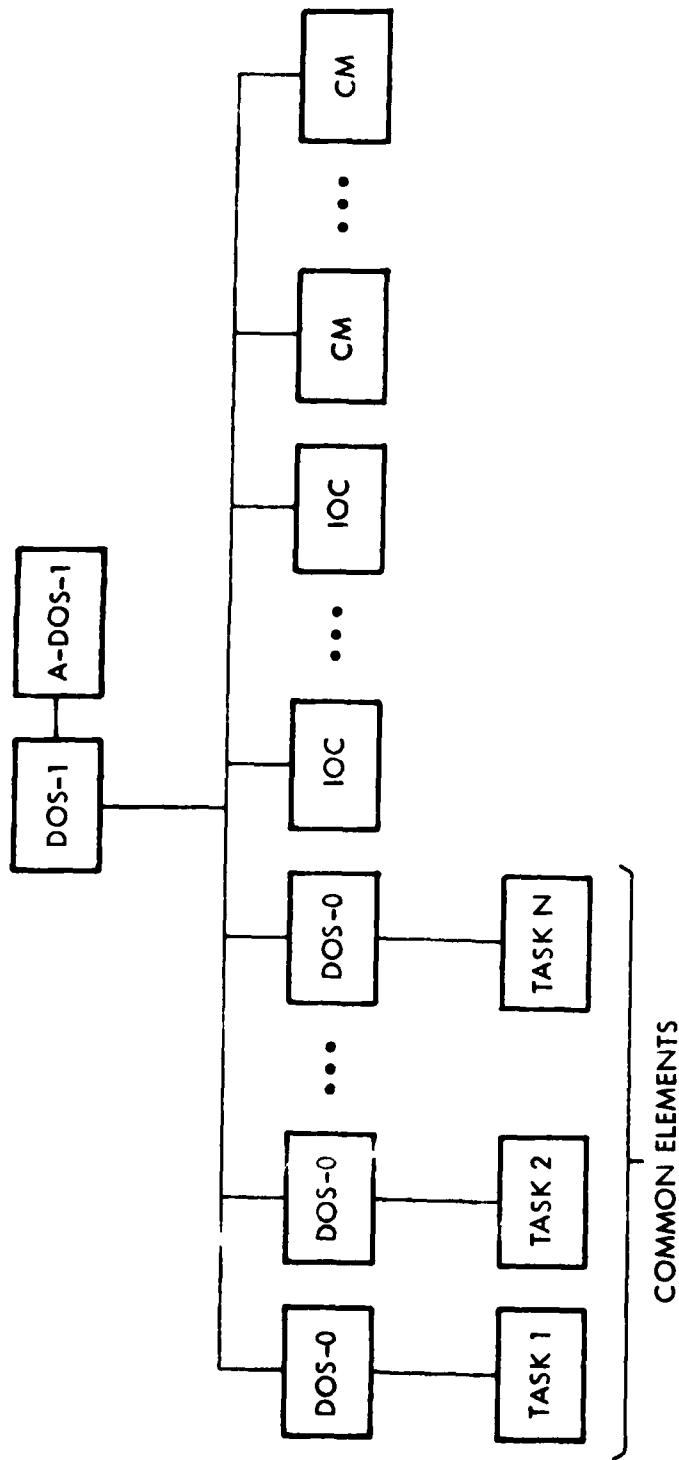


Figure 2-2. FTSP Distributed Operating System

2.3 Fault Tolerance Philosophy and Implementation

The philosophy of Fault Tolerance in FTSP is relaxed in comparison to some fault-tolerant computers. If a fault occurs, it will "eventually" be detected and removed from the system. In the meantime, erroneous results may be produced. Obviously, this philosophy would be unacceptable in some applications (e.g., control processors), but is adequate for most signal processing purposes.

Features have been built into hardware and software of the Fault Tolerant Signal Processor to aid in fault detection. The operating system isolates faults to a replaceable unit (which may be an element or a bus).

Since the reliability of a network is limited by that of its branches, special attention was given to the fault tolerance of the inter-element busses. Full handshaking (data ready, acknowledge) is performed on each word transmitted. Word-by-word parity checking helps to insure data integrity on the bus. In addition, various watchdog timers monitor all bus activity to detect timing problems.

Since more than one element may attempt to use a bus at the same time, a method of distributed arbitration is employed to decide which retains control. The virtual address of each element determines its priority on the bus (77_8 = highest, 00 = lowest). Arbitration causes lower-priority elements to yield to the highest priority element. If this process takes too long, or is otherwise thwarted, a watchdog timer detects the fault condition.

Fault-detection hardware in the CPU includes automatic checks for illegal use of privileged instructions, invalid instructions, and unauthorized use of protected memory. In addition, an extensive set of microprogrammed diagnostics is executed at power-up and after a reset condition. Under limited conditions, DOS-0 may also invoke the diagnostics and report the results to DOS-1.

Two higher-level fault detection techniques are implemented by DOS-1: 1) status polling, and 2) spare rotation. At a programmable rate, DOS-1 polls all virtual addresses with a "status request" message. No response or a reported error will cause DOS-1 to reset the element controller (e.g., DOS-0). All possible addresses are polled to help discover newly-inserted elements; this feature allows the processor to be repaired without shutdown.

Spare CEs may be assigned a self-diagnostic task to thoroughly test the hardware and firmware. By periodically rotating spares and active elements, DOS-1 insures that all CEs are checked out, providing a means for detecting subtle faults. An algorithm has been implemented which permits spare rotation without loss of data.

2.4 FTWRP Hardware

The block diagram in Figure 2-3 illustrates the FTWRP hardware and its interconnection to the existing Pulse-Pair Processor and PPP Recorder Encoder. A 30-inch rack-mounting card rack with 16 slots accommodates the three basic fault-tolerant signal processor card types:

- 1) Common signal processing Element (CE)
- 2) Input/Output Controller (IOC)
- 3) Common Memory (CM)

and to interconnects them via the dual bus system. Two IOCs and six CEs (five for processing and one spare) are provided so that eight spare slots are available for future expansion. No CMs are required in presently-envisioned applications. Power supplies, mounted in the 30-inch rack on rails, provide adequate + 5 v. 1t power for the eight cards and a test panel, and adequate ± 15 volt power for eight additional CEs. IOCs do not require ± 15 volts.

An Intecolor 8032 intelligent terminal (desk top computer), which has built-in dual floppy disk drives, is interfaced through its optional 24-bit port to the redundant bus via a Terminal Interface Element (TIE). This configuration allows the Intecolor, which serves as the FTWRP control console and fault status display, to appear as an element in the same cluster as the processing CEs. DOS-1 was recoded to reside in the Intecolor, which also has its own version of DOS-0. The terminal also serves as the system

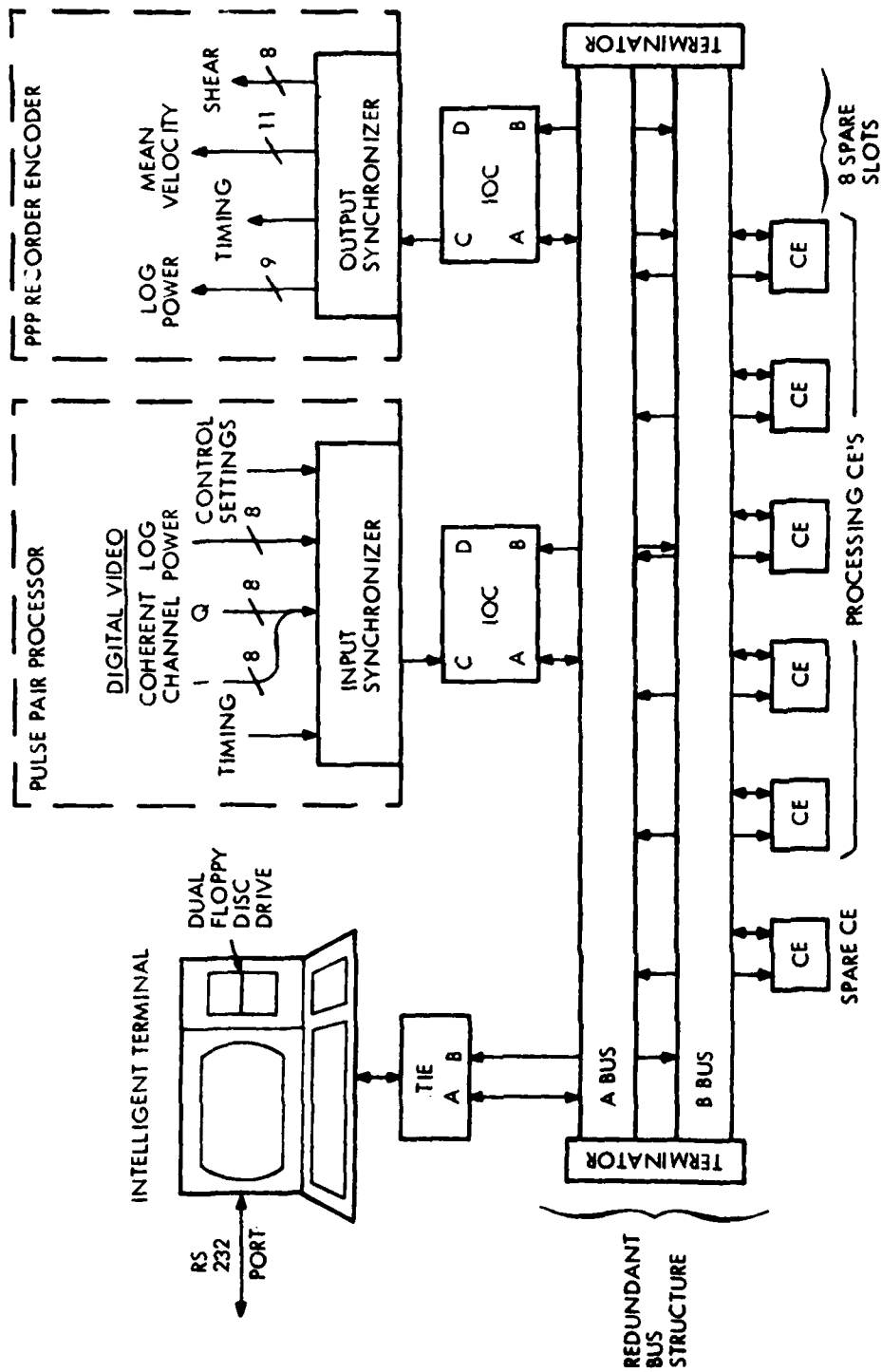


Figure 2-3. FTWRP Hardware Block Diagram

Common Memory (programs and tables are stored as pages on the dual floppy disks), and has a serial port for occasional communication with a CYBER 175 on which program development is accomplished. The existing PPP is used to provide automatic gain control and clutter cancellation. Digitized coherent channel I and Q video and log power pass into the FTWRP through the INPUT SYNCHRONIZER and its associated IOC which distributes data among the processing CEs. Following processing, the output data is collected by another IOC and passed through the OUTPUT SYNCHRONIZER to other equipment for recording or display. Section 4 presents detailed descriptions of each of the major hardware components of the FTWRP system.

2.5 FTWRP Software

The FTWRP software consists of three major parts: 1) the operating system (DOS-0 and DOS-1), 2) the FTWRP command processor (Intecolor-resident), and 3) the signal processing user programs (CE-resident).

2.5.1 Operating System Software

As stated above, the signal/data processor operating system is a two-level, distributed operating system (DOS) responsible for managing the operation of all elements within the system. The two levels of DOS correspond to the individual element (CE) level (DOS-0) and to the system level (DOS-1). Programs executing in a CE are referred to as tasks and are identified to all elements of the system by virtual address. All non-CE components also possess a virtual address to permit a uniform mechanism for communications. Tasks form the computational resource for system execution of signal processing functions. The two levels of DOS are concerned with managing the execution of tasks within each CE (DOS-0) and with the assignment and fault monitoring of tasks to available CEs (DOS-1).

2.5.2 FTWRP Command Processor

The Intecolor serves as the manual control panel for the FTWRP, by which various system parameters are communicated to the applications software. The specific nature of these parameters is dependent on the application, and is therefore described further in Sections 2.5.3 and 3.2.

In addition, the Intecolor software is responsible for reporting FTWRP system status to the operator. This status includes the socket address of each active element, its card type (CE, CM, or IOC), its current virtual address, and status. The Intecolor itself appears in the status display as a CE with socket address 0 and virtual address 77 (octal).

2.5.3 Signal Processing Application Programs

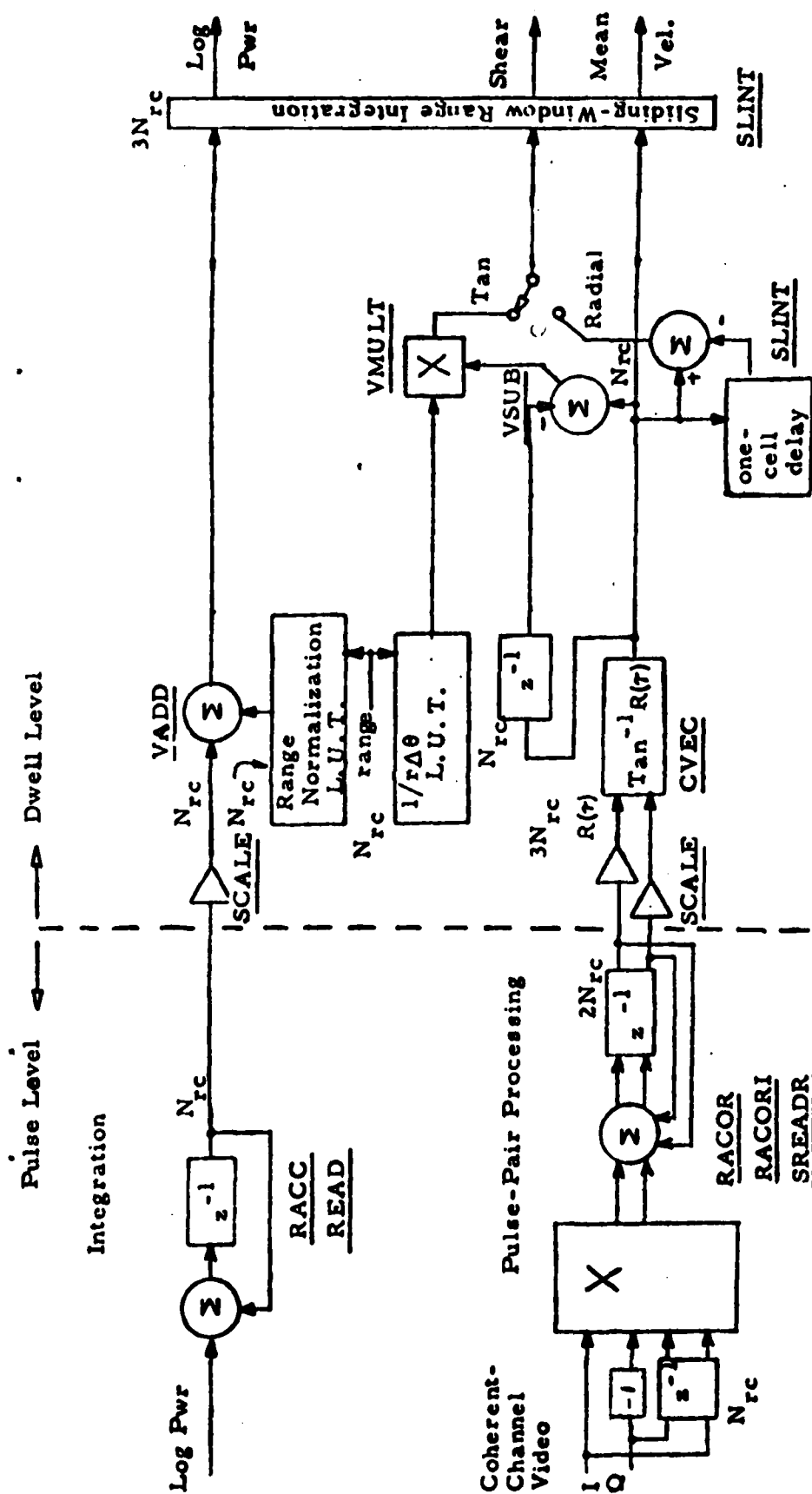
Two applications of FTWRP are available: one for the traditional radar pulse sequence and another for a special "dual wavelength" sequence developed by AFGL for range ambiguity resolution.

2.5.3.1 Continuous Pulse Sequence

In Figure 2-4, a processing flow diagram for the continuous pulse sequence is presented. In a traditional hardwired signal processor such as the existing PPP, this diagram would also represent the actual hardware. In the programmable distributed Digital Fault Tolerant Signal Processor, however, the hardware organization bears no resemblance to processing flow. Each of the processing CEs receives the same program but uses different segments of the range-addressed look-up tables. Each CE independently processes its assigned group of contiguous range cells. Prior to Range Integration, a small number of edge-cells is exchanged among CEs. The IS and input IOC distribute data among CEs and the OS and output IOC collect data upon completion of processing.

Extensive use is made of the 16K-word memory in each CE; in Figure 2-4, the buffers required for each processing function are sized in terms of the number of range cells processed (N_{rc}) for all CEs. A total of $14 N_{rc} = 14,336$ words is needed for $N_{rc} = 1024$, but since this memory is divided among five CEs, only about 25% of the available user memory area is filled.

The required signal processing macroinstructions, which were custom-developed for FTWRP (except READ and SREADR) are indicated in Figure 2-4 along with their associated processing functions. Functions to the LEFT of the dashed line ("pulse-level" processing) are especially time-critical since they are performed following each radar pulse. For this reason, RACC and RACOR received special attention in terms of optimization of execution



NOTES:

1. Memory Requirements Shown in terms of Number of range-cells processed (N_{rc})

Total = 14 N_{rc}

2. Signal Processing or Operate instructions used are shown underlined and in caps.

Figure 2-4. Processing Flow Diagram for Continuous Pulse Sequence

time. Instructions READ, RACORI and SREADR are used only following the first pulse since they eliminate all past history from the accumulators.

The SCALE instruction normalizes accumulator contents with respect to the selected number of pulses integrated and transfers their contents to other buffer areas. This process is analogous to the "dump" operation in the hardware PPP which transfers information into holding loops. The remaining operations, also performed only once per dwell ("dwell-level" processing) are executed using various combinations of minicomputer and signal processing instructions. The most noteworthy of these is CVEC, which computes the inverse tangent of the autocorrelation function for each range cell. Though not needed for the specified outputs, CVEC also computes magnitude which might be useful if, for example, it is decided to add a width output alternative as a later refinement.

The user programs run in a two-level interrupt scheme. While dwell-level processing is being executed, incoming messages are examined by DOS-0 in each CE. If the message is from DOS-1, then DOS-0 takes control and responds accordingly. For example, DOS-1 may wish to sense the status of the CE. If, on the other hand, the message is from the Input IOC, then it must be radar data so DOS-0 immediately relinquishes control to the pulse-level user program which runs in a privileged mode and cannot accept other messages since processing is done from the CE's receive buffer. When the buffer's contents have all been processed, control returns to the dwell-level where execution resumes. Another possible input message is the result of another CE transmitting a small number of range cells for range integration. In this case, DOS-0 returns control back to the dwell-level which can then complete all processing for that dwell.

2.5.3.2 Dual Wavelength Sequence Application

In a new dual-wavelength scheme being planned for implementation by AFGL (see Figure 2-5), pulses at frequency F_1 are transmitted at a uniform PRF while pulses at F_2 are transmitted at PRF/4. In the Dual-Wavelength user program returns from F_1 are pulse-pair processed to provide unambiguous velocity coverage of $\pm \text{PRF}/2$, while F_2 pulses (from a separate receiver channel)

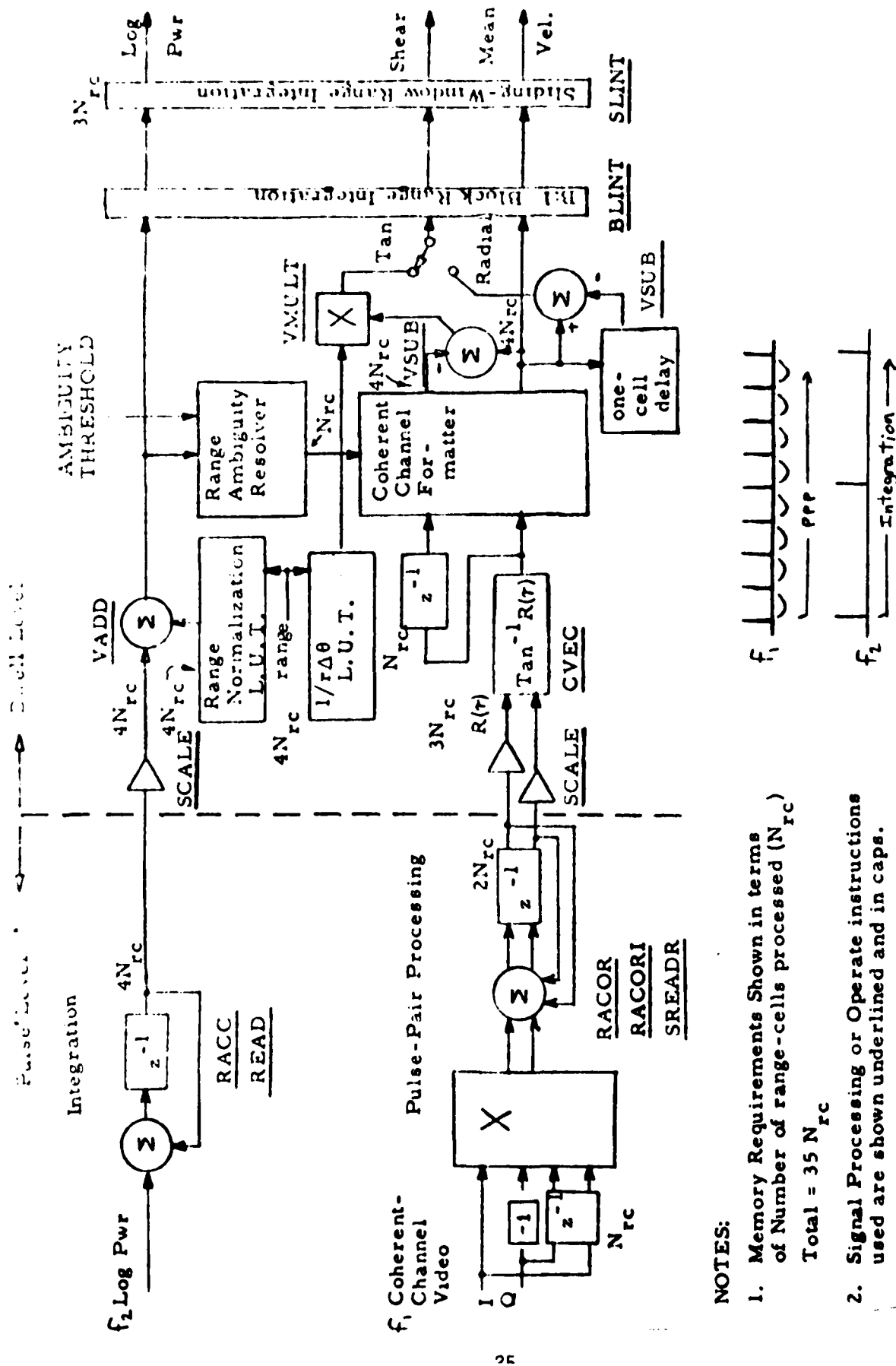


Figure 2-5. Processing Flow Diagram for Dual Wavelength Mode

are block-integrated to provide four times the unambiguous range coverage for reflectivity. Range Ambiguity Resolver and Coherent Channel Formatter software then unscrambles the range-ambiguous F_1 doppler data based on F_2 reflectivity information. Written using the minicomputer instruction set for maximum user flexibility, these functions use log power (reflectivity) data which is unambiguous over four times the range of the coherent data to format velocity and shear into an apparently unambiguous range extent. Since this process results in far more range cells than the Encoder can handle, a B:1 block range integration is performed prior to output. Normally B is chosen as 4, but other possibilities allow finer resolution with non-standard range-cell scaling. Additionally, sliding window range integration can be performed if desired, just as in the continuous pulse case.

REFERENCES

- 2-1. Stiffler, J.J., "On the Efficacy of R-on-M Redundancy", IEEE Transactions on Reliability, Vol. R-23, No. 1, April 1974, pp. 37-43.

3. FTWRP SYSTEM USAGE

The following sections describe the use of the FTWRP system and the command interpreter. Section 3.1 deals with system startup, and gives a step-by-step instruction outline on how to bring up the system from a powered-down state. Section 3.2 describes the command interpreter and required command formats. System power-down is discussed in detail in the last section.

3.1 FTWRP System Startup

To bring the system up from a cold start, follow the simple instructions below:

1. Turn on the Intecolor (switch located in rear of terminal).
2. Turn on FTWRP power.
3. Reset all cards by setting RESET switches on front panel of FTWRP to down position and then returning them to their normal (up) position.
4. Turn on power to Pulse-Pair Processor and PPP encoder/decoder.
5. Turn on power to Scan Converter.
6. Insert diskette labelled "FTWRP" in drive 0 (lower drive) on the Intecolor.
7. Insert diskette labelled "FTWRPAGES" in drive 1 (upper drive) of the Intecolor.
8. Type (ESCAPE) P.
9. Type (ESCAPE) D.
10. Type RUN IDOSO (RETURN).
11. Type in commands as desired (see Section 3.2).
12. Type PPP to begin processing.

The Intecolor should begin accessing the drive 1 diskette repeatedly as the Common Elements request pages to load the applications software. Approximately 16 pages per CE will be loaded before the system will actually begin to perform the task.

3.2 IDOS-1 FTWRP Command Interpreter

The interpreter has a self-contained command parser and built-in error checking to provide a simple user interface. This section describes each of the commands currently supported, and gives specific format requirements.

After each command is a parameter list--required parameters are enclosed in square brackets '[' and ']'. Optional parameters are enclosed in parentheses '(' and ')'. Default values for optional parameters are given in the description that accompanies each command.

Parameters enclosed by curly brackets '{' and '}' denote that exactly one of the values listed must be included.

Each command must be followed by at least one space. Commands are truncated to 5 characters, and all following characters are ignored.

Parameters are separated by either a comma ',' or by one or more spaces ' '. Number parameters may be in octal, hex, or decimal, and bases may be interchanged within commands. Numbers of different bases are entered as follows:

DECIMAL	NUMBER	EXAMPLE:	678
			-429
OCTAL	' NUMBER	EXAMPLE:	'357
			'0103
HEXADECIMAL	H NUMBER	EXAMPLE:	HA94E

In addition, some parameters require floating point format for entry. Floating point numbers must be in decimal, and are of the form

XXXXXX.YY

where XXXXXX is any number from -32768 to +32767

and

YY is any one or two digit number up to 99. Leading and trailing zeros need not be included. If the fractional part is zero, the decimal point need not be included.

EXAMPLES:

34.67

-45.1

109

The command descriptions are given in Table 3-1.

3.2.1 FTWRP Parameters

System parameters are entered using the SET command (see Table 3-1). The various parameters and their possible values are listed in Table 3-2. Some parameter values are affected by others, and there are limitations on the combinations of values. For example, the default scale constant (SCL) is determined by the current setting of the number of pulses integrated (NSI) according to Table 3-3.

If the shear flag (SHR) is set to RADIAL, the SHRLUT parameter, which is useful only to tangential shear computation, is ignored. Also, many parameters are used only during range ambiguity resolution, which is not performed except in the Dual-Wavelength application. These include BLW, PRETHR, and ZTH.

Finally, there are limitations on the combination of values for the number of range cells per pulse (NRC), the range cell size (TP), and the number of pulses integrated (NSI). Table 3-4 lists these in detail.

Table 3-1. FTWRP Command Description

<u>COMMAND</u>	<u>DESCRIPTION</u>
RES [VA]	Reset card with virtual address VA
DIR (DRIVE#)	Display directory of disk drive DRIVE#. Default drive is MD1:
SAV [FILENAME](.EXT)(;VR)(MEMSPEC)	Save a file onto disk with name 'FILENAME.EXT;VR' If the memory spec is not included, page 0 will be saved as 'FILENAME' onto drive MD1:
LOA [FILENAME](.EXT)(;VR)(MEMSPEC)	Same as save command, except file will be retrieved from disk and placed in specified area of memory. Default memory spec is page 0
RADIX OCTAL, HEX	Sets the radix for output to the specified value.
CYB (BAUD)	Sets the baud rate as specified in BAUD, then calls the utility routine CYBER which interfaces the Intecolor
BAUD:	
0-7 = 300 baud	
8 = 1200 baud	
16 = 2400 baud	
32 = 4800 baud	
64 = 9600 baud	
	to a modem connected to the RS-232 port. Control is regained when the DELETE key is depressed The default baud rate is 300 baud.
ERA	Erase the Intecolor Display

INI	Initializes the IDOS-1 tables (PGMAP, SYSTBL, etc) and returns IDOS-1 to the initialization state
TSK [VA], [TSKID]	Load task into CE with address VA. 'TSKID' is used to compute the Load Map Page number.
ST [VA], [NEWVA](,STARTADDR)	Start CE executing task under address 'NEWVA' and the specified start address. If start address is omitted, the address in the task prologue will be used.
CLR	Clear the Intecolor display, then repaint the system status on the screen.
PRINT ON,OFF	This turns on the Integral Data Systems printer handler and causes all output to the screen to be printed. This handler may not work for all printers.
DSPP [PAGENUMBER]	Displays the desired page in octal or hex format, depending on the current radix
DSPM [VA][,STARTADDR](,ENDADDR)	Displays the contents of CE memory, starting at STARTADDR and ending at ENDADDR. If ENDADDR is omitted, 240 words will be displayed. In any case, no more than 240 words will be displayed.

MODP [PAGENUMBER],[ADDR1=DATA1](,DATA2)(,DATA3)...(ADDR2=DATA1)(,DATA2).

Modifies the desired page by replacing the current contents at ADDR1,etc with DATA1,etc. Consecutive addresses need not be explicitly entered. If page 0, 1, or 2 are modified, only the RAM-resident versions are modified. Otherwise, the new copy is written to disk, and the old copy is destroyed.

MODM [VA],[ADDR1=DATA1](,DATA2)...(ADDR2=DATA1)(,DATA2)...

Same as MODP except destination is the specified memory address in CE with virtual address VA.

MSG [DESTVA],[SRCVA](,MSGCODE)(,WORDCT)

Sends the contents of page 0 to DESTVA, creates a header word with SRCVA as the source address and message code MSGCODE. Exactly WORDCT words will be sent (not including the header or wordcount words) If WORDCT is omitted, 240 words are transmitted.

SVA [VA]

Changes the virtual address of the Intecolor to VA.

BUS {A,B,ALT}

Selects the bus over which all subsequent transmissions will be sent. ALT signifies that busses will alternated.

CMR [VA],[PAGENUMBER]	Sends a CM read request to VA, for page PAGENUMBER. The result, when it is received, will be displayed on the screen.
CMW [VA],[PAGENUMBER]	Sends a CM write request to VA to write the contents of page 0 as page PAGENUMBER.
SCH [TSKNO],[VA]	Enters the task id TSKNO onto the DOS-1 task queue, and VA onto the virtual address queue.
CON [VA]	Loads the configuration table to the CE at address VA.
TRA [VA]{ON,OFF}	Turns trace feature in or out at VA on or off. For this command to work properly, trace interrupts must have already been enabled in user task prologue.
TSP [VA]	Sends a suspend task command to the CE at VA.
TRS [VA]	Sends a resume task command to the CE at VA. The CE must have previously been sent a suspend task message.
STO	Saves the current task queue and related variables on drive MD1: as file 'TASK S.001'

CTQ	Clears the task queue and virtual address queue.
ABORT	Aborts the pulse pair processing tasks and returns the system to the idle state.
CONT	Restarts the Pulse-Pair processing tasks after an ABORT command.
PPP	Begins the System Startup procedure that performs Pulse-Pair processing.
SET [PAR1=DATA1](,PAR2=DATA2)... (PARn=DATAn)	Modifies the parameter table as instructed in the operands (see Table 3-2). If the system is idle, nothing else happens. Otherwise, the modified parameter list is sent to each signal processing CE.

Table 3-2. 'SET' PARAMETER MNEMONICS

<u>MNEMONIC</u>	<u>DESCRIPTION</u>
SHR	Shear flag: R = Radial shear processing T = Tangential shear
SHRLUT	Tangential shear Look up table load map page number. Not applicable if shear flag = Radial.
NRC	Number of Range Cells Processed: 256 512 768 1024
NSI	Number of Pulses Integrated: 16 32 64 128 256
SLW	Sliding Window Integration window size: 0 - 16
SCL	Scale constant (if different from default): 1 - 256
BLW	Block Integration window size (for dual wavelength only): 1 2 4

Table 3-2. 'SET' Parameter Mnemonics (con't)

<u>MNEMONIC</u>	<u>DESCRIPTION</u>
PRETHR	Pre Range Ambiguity Resolution Reflectivity Threshold (dual wavelength only): 0 - 100
ZTH	Post Range Ambiguity Resolution Reflectivity Threshold (dual wavelength only): 0 - 100
RNORM	Range Normalization processing: ON OFF
TP	Range Cell size (in microseconds): 1 2
STPOLL	Status polling frequency (in seconds): 1 - 255
SPRROT	Spare Rotation parameters (2): ON, frequency (in seconds, 1 - 2 55) OFF
IOC	Number of IOCs required for processing: 0 - 2
CE	Number of CEs required for processing: 0 - 5
STPMAX	Number of unanswered status polls allowed before reconfiguration is invoked: 1 - 255
DEGRAD	System Degradation flag: ON (system will degrade) OFF (system will do nothing)

Table 3-3. Scale Constant Default Determination

<u>NSI</u>	<u>SCL (Default)</u>
32	8
64	4
128	2
256	1

Table 3-4. System Limitations on NRC, TP, and NSI.

('X' denotes illegal combinations)

		NRC			
	NSI	256	512	768	1024

	32	X	X	X	-
TP =	64	X	X	?	-
1 usec	128	X	X	-	-
	256	X	X	-	-

	32	-	-	-	-
TP =	64	-	-	-	-
2 usec	128	-	-	-	-
	256	-	-	-	-

See Appendix D

3.3. FTWRP System Shutdown

To shut the FTWRP system down, follow the steps below:

1. Type ABORT (RETURN) to stop the signal processing tasks.
2. Hit the CPU RESET key on the Intecolor keyboard.
3. Turn off power to Intecolor, SCRM, PPP, PPP encoder/decoder, and FTWRP (in any order).

4. DETAILED HARDWARE DESCRIPTION

4.1 Common Element

4.1.1 General Description

The Common Element (CE) is a complete microprogrammed 16-bit computer on a card, including I/O ports, memory, ALU and control sequencer. Figure 4-1 is a block diagram of the CE. The CE instruction set is a superset of the Raytheon RP-16 microcomputer instruction set with signal processing macro instructions; e.g., complex matrix multiply, CVEC etc. Several hardware features have been incorporated in the CE to permit 32 or more elements to operate on common busses and to make software independent of hardware assignments. This section discusses the CE in general, and gives an Equipment Specification of the Mark I CE.

Two I/O ports connect to identical 16-bit bi-directional busses which provide redundant paths for macro-program loading into CEs and for communication of data both among CEs and with other memory or I/O units connected to the busses. The control lines are used for bus control and arbitration. Bus data rates are dependent on many physical parameters, such as the distance between source and destination. However, the average data rate is approximately four 16-bit words per microsecond (4 million words/sec).

The Random Access Memory, used for storage of data and macroprograms, has dimensions of 16K words by 16 bits. Although it is dynamic MOS memory and requires refresh, this requirement is made transparent to macroprograms by inclusion of an address multiplexer and refresh timer which ensure adequate refresh through a micro-coded routine. A basic read or write RAM cycle requires two micro cycles, but for access to sequential addresses within 128 word "pages", only one micro cycle is required.

The structure of the Arithmetic Unit has been optimized for signal processing operations such as complex matrix multiplication, convolution, Fast-Fourier Transform, etc. The ALU is a Schottky-bipolar bit slice unit which can implement typical minicomputer operations such as addition,

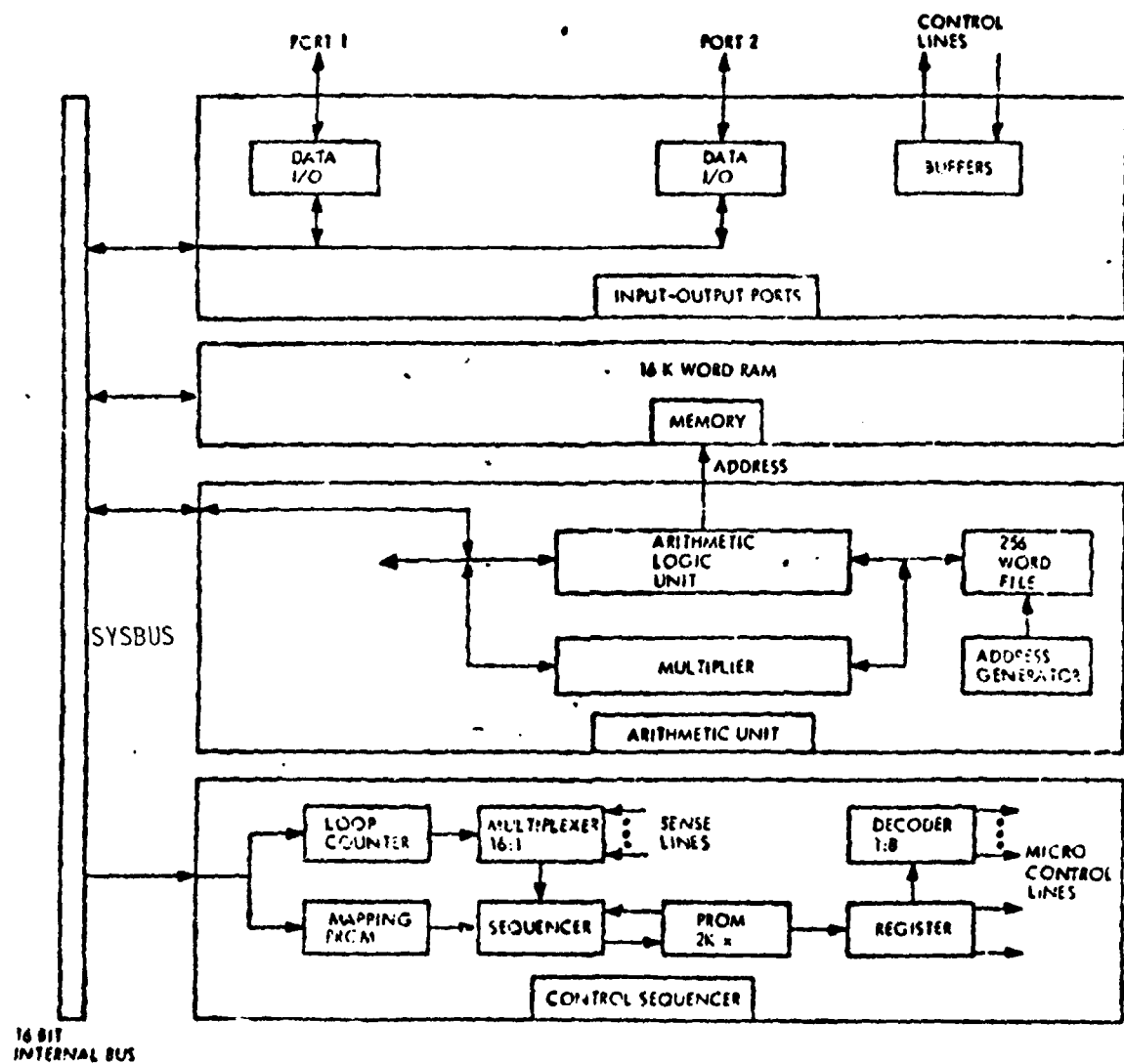


Figure 4-1. Simplified Common Element Block Diagram

logical functions, shifts, and sequential multiply or divide. The ALU also produces an address for the RAM by performing arithmetic operations on internal registers so that various addressing modes can be accomplished. Most operations can be performed in one micro cycle; in fact, many compound operations such as simultaneous address and data computations are possible. Multiplication and division, however, require about one micro cycle per bit. For this reason, a high-speed parallel multiplier which can perform a 16 x 16 signed two's complement multiply in just one micro cycle was added. The 256-word cache memory serves as a high-speed register file for storage of intermediate results of data processing algorithms. Read or write accesses can be done in one micro cycle with sequential locations accessed by an address generator under micro-program control.

4.1.2 Specification - Common Element Mark I (Part No. 977725)

4.1.2.1 Description

The CE is a complete microprogrammed processor on a card, having ALU, memory, dual I/O circuitry, and a control sequencer. The CE is designed to function as one of many programmable elements including spares collectively communicating via dual-independent high-speed data busses to form a fault-tolerant signal processor wherein the failure of one bus or any CE can be tolerated. The CE is optimized for high-speed signal processing by virtue of its architecture which includes a hardware multiplier so that multiply/add operations can be accomplished in one microcycle.

4.1.2.2 Specification Summary

4.1.2.2.1 Memory -

o RAM for program and data storage:	16,384	16-bit words
o RAM for register sets and file:	256	" "
o RAM for input buffer:	256	" "
o RAM for output buffer:	256	" "
o ROM for DOS-0	4,096	" "

- o ROM for Microcode: 2,048 80-bit "
- o ROM for Instruction Decode 1,024 16-bit "

4.1.2.2.2 ALU -

- o 16-bit 2's complement arithmetic
- o 2 working registers
- o 2 memory address registers
- o separate 16 x 16 hardware multiplier
- o 250 nsec maximum cycle time

4.1.2.2.3 I/O -

- o Two independent asynchronous parallel ports, each with distributed bus-arbitration logic
- o 16-bit width plus one parity bit
- o Message blocks up to 256 words long
- o Header string for message routing and identification
- o Four MHz typical bus data rate
- o Up to 32 elements per bus
- o Built-in control line timing and sequence monitoring

4.1.2.2.4 Software -

- o ROM-resident element-level distributed operating system DOS-0 which coordinates CE operation and handles allocation of CE resources, I/O, and fault monitoring
- o 30 mini-computer instructions; e.g., ADD
- o 24 operate instructions used by DOS; e.g., WRITE
- o 12 powerful signal processing instructions; e.g., RACOR which computes autocorrelation for many range cells and is the heart of the Pulse Pair Processing algorithm
- o Eight sets of eight general-purpose registers

4.1.2.2.5 General -

- o Construction: wire-wrap, plug-in, dual-inline 0 to 70⁰ C Integrated Circuits
- o Size: 16.3125 x 13.8125 (including connector) x 1.5 inches
- o Power Requirements:

	<u>Typ.</u>	<u>Max.</u>
+ 15V \pm 5%	.1A	.6A
- 15V \pm 5%	.025A	.04A
+6.3V \pm 5%	15A	20A

- o Built-in power controller monitors and regulates voltages, provides proper sequencing at turn-on, allows addition of redundant power sources, and allows faulty CEs to be powered-down.

4.1.3 Detailed Hardware Description

The following sections describe the Common Element hardware and timing in more detail. The discussions emphasize six major units of the CE: I/O transceiver, Arithmetic Unit, Control Sequencer Unit, RAM Interface Unit, Timing Generator, and Power Controller.

4.1.3.1 Input/Output Transceiver

The Bus Transceiver circuit of the Common Element (CE) is used by the CE to transmit data to and receive data from other elements in its cluster by means of either of two 16-bit wide open collector buses. Data is transferred in blocks of up to 256 words. Transmitting and receiving are under control of independent control circuits. Since several transceivers may attempt to use the bus at the same time, a system of arbitration is employed to decide which transmitter will control the bus. Figure 4-2 is a detailed block diagram of the CE I/O and Figures 4-3 and 4-4 are timing diagrams showing the transmit and receive operations.

4.1.3.1.1 Inputs from CE

1. IOBUFOEC
 - o Outputs receiver RAM to SYSBUS when low.
 - o RAM address advances on pos. edge of CKO.
 - o Must be high when receiver is idle.
2. IOSTATOEC
 - o Outputs status to SYSBUS when low.
3. SEL BUS
 - o Selects bus according to state of SBO (H = B, L = A).
 - o Clocks on pos. edge.
 - o Must stay high when transmitter is triggered.

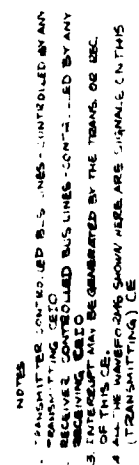


Figure 4-3. Transmitter Timing.

4. TXRST

- o Resets receiver to idle from any state when low.
- o Block size high when receiver is idle.

6. LTOADR

- o Loads CE address from SYSBUS on pos. edge.
- o New address will be used immediately by receiver.
- o Transmitter will use newly loaded address on first TST after loading.
- o Do not load new address while transmitter is triggered.
- o This address does not effect the data block header word.

7. TST

- o Starts transmit sequence when low.
- o Uses latest loaded address for arbitration.
- o May be used to reattempt transmission of a block previously loaded into transmitter RAM if no TXRST is given.

8. TRBULD

- o Loads one word per CK0 cycle from SYSBUS to transmitter RAM when low.
- o Loading does not have to be continuous.
- o Load only when transmitter is idle.

4.1.3.1.2 Arbitration

Each transceiver has a 6-bit address loaded from the SYSBUS into a register on command of the CE. The address is divided into two 3-bit groups (MSB and LSB), each of which goes to a one-of-8 decoder to produce the 16-bit arbitration code. When the bus becomes unoccupied, one or more previously-triggered transmitters may occupy the bus simultaneously, each pulling a pair of data lines low according to its arbitration code. The code seen on the bus is the wired-OR of all the codes occupying the bus. The presence of two or more transmitters on the bus necessarily implies that more than two lines are pulled low. The presence of higher priority transmitters on the bus causes lower priority transmitters to drop off. When a transmitter sees that the only others on the bus are lower priority, it waits for them to drop off. When its two bits are the only ones pulled low, it has won the arbitration and proceeds to transmit its data block.

4.1.3.1.3 Destination decoding

After winning arbitration, the transmitting element outputs the first RAM word to the bus and pulls the $\overline{\text{READY}}$ control line low. The first word is a block-identifying header which includes the six-bit card address of the element which is to receive the block. At the time of the first negative-going edge of $\overline{\text{READY}}$ in the transmit cycle, every element in the cluster strobes the output of its destination decoder to see if the six destination bits of the header match its card address bits.

4.1.3.1.4 Data transfer

The receiver which is the intended destination of the block may respond in either of two ways. If it already has a data block in its RAM which has not been serviced by its CE, it sets BUSY and ACK low. If its receiving RAM has been released by its CE, it leaves BUSY high, clocks in the header and sets ACK low. Transmitter and receiver proceed to transfer the data block. The transmitter indicates the presence on the bus of the next valid data word by the falling edge of $\overline{\text{READY}}$ and the receiver acknowledges receipt of the word by the falling edge of ACK.

To determine when the transfer is complete, the transmitter compares the RAM address with the output of a counter which counted the number of write pulses during the last RAM loading period.

4.1.3.1.5 Parity

The transmitter generates odd parity on the data, and the receiver checks this parity bit against the received word. For an error on any word except the header, the receiver sets BUSY low at the time of the falling edge of ACK. Once a parity error has been received, the receiver latches BUSY low for the remainder of the transfer. A properly operating transmitter immediately terminates the transfer.

4.1.3.1.6 Transmitter controller outputs

1. $\overline{\text{AX}}$ - Parity generator output gate, A bus (L)
2. $\overline{\text{BX}}$ - Parity generator output gate, B bus (L)
3. $\overline{\text{A BUS ENBL}}$ - A bus data transmitter enable (L)
4. $\overline{\text{B BUS ENBL}}$ - B bus data transmitter enable (L)

5. $\overline{\text{CODE OE}}$ - Arbitration code output enable to T register (L)
6. $\overline{\text{DATA OE}}$ - RAM output enable to parity gen, (T) register (L)
7. $\overline{\text{TENBL}}$ - Bus transmitter enable: Ready, Occupied, Parity (L)
8. TREGGK - Clock for T register (POS EDGE)
9. TCCK - RAM address counter clock (POS EDGE)
10. $\overline{\text{TCRST}}$ - RAM address counter direct reset (L)
11. $\overline{\text{TCLOAD}}$ - RAM address counter synchronous load (L)
12. ARB ENBL - Arbitration decoder enable, MSB's (H)
13. IOINT - Interrupt to CE (H)
14. $\overline{\text{ST1}}$ thru $\overline{\text{ST4}}$ - Transmitter state (LOW TRUE).

4.1.3.1.7 Receiver controller outputs

1. RCCK - RAM address counter clock (POS EDGE)
2. $\overline{\text{RCRST}}$ - RAM address counter direct reset (L)
3. REOBCK - Block size register clock (POS EDGE)
4. $\overline{\text{R WRITE}}$ - RAM write enable (L)
5. $\overline{\text{ST5}}$, $\overline{\text{ST6}}$, $\overline{\text{ST7}}$ - Receiver state
6. $\overline{\text{ICA}}$ - A register output enable to RAM (L)
7. $\overline{\text{ICB}}$ - B register output enable to RAM (L)
8. A REG CK - A register clock (POS EDGE)
9. B REG CK - B register clock (POS EDGE)

4.1.3.1.8 Status word

The transceiver outputs a 16-bit status word to the SYSBUS on command of the CE. Included in the status are the transmitter state, receiver state, last transmit bus, and the length of the last received block. Tables 4-1 through 4-4 are truth tables for the word.

4.1.3.1.9 Receiver states

Idle

No data waiting in RAM
May be receiving
Do not reset receiver when it is in this state

Parity Error A

Parity error has occurred during reception on bus A
Sender is still occupying bus

STATUS TRUTH TABLES
(At SYSBUS)

Table 4-1 TRANSMITTER STATE

SB4	SB3	SB2	SB1	STATE
H	H	H	H	TRIGGERED
H	H	L	H	BUS BUSY*
H	L	H	H	ARB FAULT*
H	L	L	H	REPLY FAULT*
L	H	H	H	REC BUSY*
L	H	L	H	PARITY ERROR*
L	L	H	H	TIME FAULT*
L	L	L	H	DONE*
L	L	L	L	IDLE

*(interrupt)
No other states of SB1 thru SB4 occur

Table 4-2 LAST TRANSMIT BUS

SB0	BUS
L	A
H	B

Table 4-3 RECEIVER STATE

SB7	SB6	SB5	STATE
H	L	L	IDLE
H	H	L	PARITY ERROR A
H	L	H	PARITY ERROR B
L	H	L	FAULT A*
L	L	H	FAULT B*
L	L	L	FULL*

*Interrupt
No other states of SB5 - SB7 occur

Table 4-4 BLOCK LENGTH

SB15	SB14	SB13	SB12	SB11	SB10	SB9	SB8	LENGTH
L	L	L	L	L	L	L	L	HEADER ONLY
L	L	L	L	L	L	L	H	HEADER & ONE WORD
L	L	L	L	L	L	H	L	HEADER & TWO WORDS
o	o	o	o	o	o	o	o	
o	o	o	o	o	o	o	o	
H	H	H	H	H	H	H	H	HEADER & 255 WORDS

Parity Error B

Parity error has occurred during reception on bus B
Sender is still on bus

Fault A

Incomplete block received on A
Parity error or time elapsed

Fault B

Incomplete block received on B
Parity error or time elapsed

Full

Data block received without error
Read onto SYS BUS any time before next RXRST

4.1.3.1.10 Transmitter states

Triggered

Sending or waiting to send
Will proceed to an interrupt state
TST has been received
May be reset from this state directly

Bus Busy

Unable to get bus and win arbitration in required time

ARB Fault

Too much time taken in arbitration

Reply Fault

No reply from receiver
May mean parity error on header

Rec Busy

Receiver answered "BUSY"

Parity Error

Transmission incomplete due to parity error

Time Fault

Transmission incomplete
Transmission took too much time
Receiver did reply to header

Done

Transmission completed
Must get RXRST to go to "IDLE"

Idle

Must get TXRST to get to this state
Load transmitter buffer only when transmitter is in this state
Counts block length: #Words written between TXRST and next TST
Goes to "TRIGGERED" on receipt of TST

4.1.3.2 Arithmetic Unit

The CE Arithmetic Unit (see block diagram Figure 4-5) consists of the ALU, the Multiplier, Program Status word, and Register File. The following subsections discuss each of these in detail.

4.1.3.2.1 Arithmetic Logic Unit (ALU)

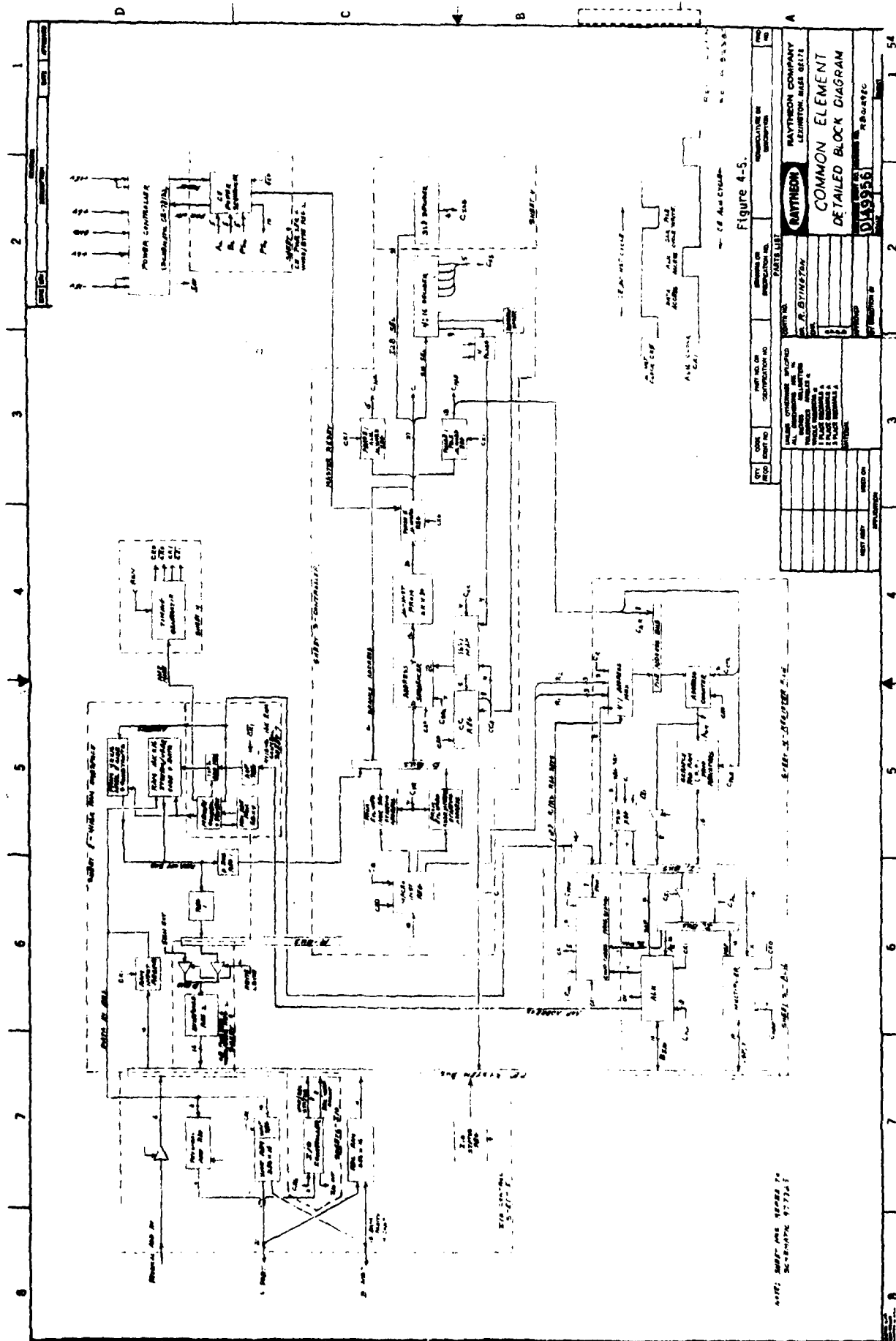
The ALU used by the Common Element is the Texas Instrument SN74S481 4-bit slice, shown functionally in Figure 4-6. The 'S481 contains many features relevant to the CE architecture, most notably four internal special purpose registers and four major data ports. The working register (WR) and extended working register (XWR) can be used separately or concatenated to form a single double length accumulator. The memory counter(MC) and program counter (PC) registers represent independent memory address generators with separate increment-by-one and increment-by-two controls. The I/O ports of the ALU chips include two data input ports, AI and BI/O (which doubles as an output port), a general purpose data out port (DOP) and an independent address-out port (AOP), permitting memory addressing without tying up the entire ALU. The many capabilities of the 'S481 are discussed in detail in Reference 4-1.

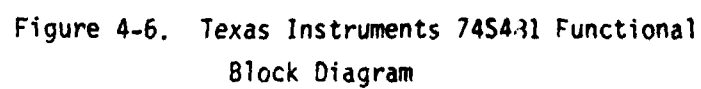
4.1.3.2.2 Multiplier

The on-board multiplier in the CE is a TRW MPY16AJ 16 x 16 multiplier array, contained in a large, 64-pin dual-inline-package. The MPY16AJ performs a 16-bit two's complement fractional multiply in about 200 ns. It is connected to the ALU in such a manner so as to permit a single cycle accumulated multiply.

4.1.3.2.3 Register file

The Common Element Arithmetic Unit has a 256 x 16 RAM with a single-cycle access capability. The first 64 locations are reserved for eight sets of eight general-purpose registers used by the data processing





instruction set. The second 64 locations are reserved for internal book-keeping in microcode, and the upper 128 locations are used as scratch-pad RAM by the signal processing macro-instructions.

4.1.3.3 Control Sequencer Unit

The Control Sequencer Unit consists of the microprogram Sequencer, the two mapping PROMs, the pipeline registers, and the condition code select multiplexer. These are each described in the following subsections.

4.1.3.3.1 Microprogram Sequencer

The microprogram sequencer is an AMD Am2910 chip housed in a 40-pin dual-inline package. The sequencer generates the next microprogram address based on various states of the machine by using a powerful set of instructions. It uses as inputs the condition code multiplexer output (as a method of conditional execution), the D port input (to which various constants can be input), and internal registers such as the microprogram counter (μpc) and register/counter (for loop counting). The AM2910 also has an on-chip microprogram stack which supports up to 5 levels of subroutines in microcode.

4.1.3.3.2 Pipeline Registers

The output of the 2Kx80 microinstruction PROM is registered to increase throughput. All 80 bits are registered on the rising edge of CK0 for initial pipelining. However, to effect the two-phase operation of the CE, all bits which control hardware that is dependent on CK1 are re-registered at the rising edge of CK1. Thus, two-phase operation is accomplished automatically.

4.1.3.3.3 Mapping PROMs

The CE contains two sets of 512 x 8 bit mapping PROMs. The first set is used to vector to microcode routines which compute the effective memory address implied in the various data processing address modes which the CE supports. Once computed, the effective address is stored in the ALU memory counter (MC) register for later use. The second mapping PROM is then used to vector to the microroutines which actually execute the opcode as required. The various instructions and addressing modes are treated in detail in the CE Programmer's Handbook (Reference 4-2).

4.1.3.4 RAM Interface Unit

The onboard memory consists of 16K words of dynamic RAM for user program and data storage, plus 4K words of PROM for operating system use. The dynamic RAMs (4116-2) have two possible addressing modes. A random memory read or write requires the row address to be latched first and then the column address. The high to low transition of the ROW ADDRESS STROBE (RAS) latches the row address. The high to low transition of the COLUMN ADDRESS STROBE (CAS) latches the column address. Page mode allows successive memory operation at any column address locations of the same row. This increases the speed by approximately a factor of 2 without an increase in the operating power. The only limitation on read or write memory operations that can be performed in page mode is the length of time the row address is valid. After 10 micro-seconds the row address must be pre-charged (i.e. after 10 μ s of page mode operations the next memory operation must be a random access to strobe both the row and column address). The page mode cycle is 170 nano-seconds minimum and the random access cycle is 375 nano-seconds minimum. With tolerance for timing this makes the page mode cycle equal to the CE cycle and the random access cycle equal to two CE cycles. The decision to use page mode or random-access mode is made in hardware, independently from microcode or software.

At the output of the RAM is a byte/word register which permits automatic unpacking of bytes with either sign extension or zero-fill options.

A hardware timer interrupts the microcode approximately every 2 ms for RAM refresh. The microcode then sequentially accesses all 128 row addresses to fully refresh the entire RAM. This dynamic RAM refresh timer also serves as the basic tick for the CE's user clock used by software.

A memory protect circuit prevents writing into privileged areas of RAM without proper authorization. Privileged areas of memory are the user task prologue (see Section 5.1.2.2), DOS-0 data memory, the unused locations from $3FFF_{16}$ to $F7FF_{16}$, and the PROM address space ($F800-FFFF$). An unauthorized write to protected RAM is changed to a read operation and a fault interrupt is generated.

4.1.3.5 Timing Generator

The Timing Generator creates the master clock signals for the entire board. The major input to the generator is the page-fault signal from the RAM interface, which stops the clock temporarily while a long random access operation takes place. The clock itself consists of two phases, CK0 and CK1 (see Figure 4-7). This effectively breaks the CPU cycle into four parts, labeled T_1 , T_2 , T_3 and T_4 .

The first part, T_1 , is used primarily for data setup on internal busses, memory address decoding, microcode instruction decoding etc, and is approximately 65-71 ns in duration. The ALU accepts its inputs and decodes its instructions during T_2 , which is 50-55 ns long. T_4 is constrained solely by the register file write operation, and is 34-37 ns in duration. The third part, T_3 , is used to ensure that all hold times and cycle time restraints are met, and is 73-80 ns long. The CE cycle time is therefore specified as 222 ns minimum, and 243 typical. The average speed of the FTWRP Common Elements is approximately 240 ns.

4.1.3.6 Power Controller

The Power Controller serves three major functions: 1) as an onboard regulator, which takes the +6V and $\pm 15V$ power bus voltages and regulates them down to +5V, +12V, and -5V required by the CE; 2) as a power sequencer, which ensures that the CE voltages are applied in the proper order to avoid blowing out any sensitive IC's (such as the RAMs); and 3) as the mechanism by which the board is reset (master reset) or powered down by executive command. The majority of the power controller is contained on the piggy-back printed-circuit board mounted at the connector end of the card.

4.2 Input/Output Controller (IOC)

The IOC module, block-diagrammed in Figure 4-8, serves as a message center with routing, control and error-checking functions. Data enters as word-serial packets at a rate of 4×10^6 16-bit words per second through one of four ports. Each packet is loaded by the Receive Controller into one of two RAMs. Processing such as header reordering for intercluster transfers is performed by the Block Controller,

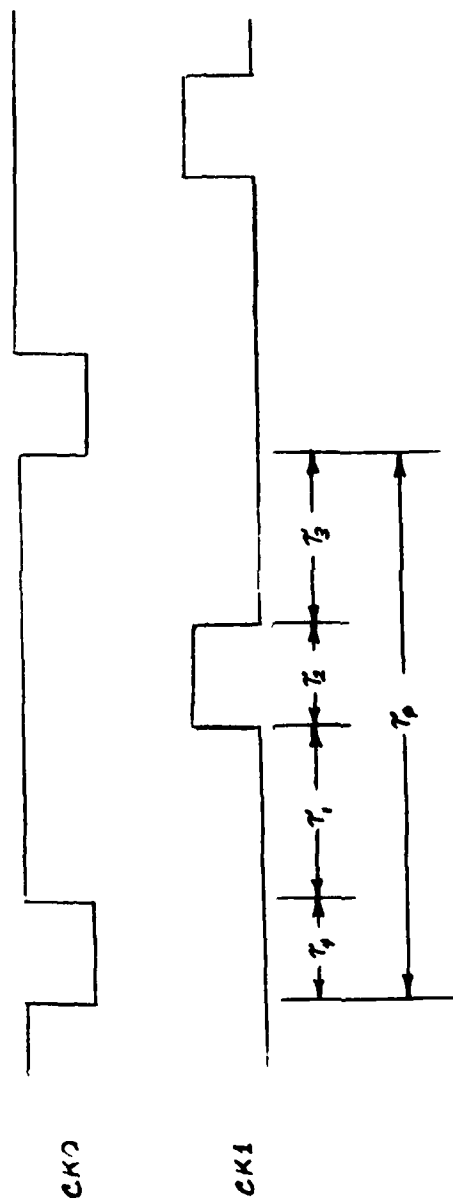


Figure 4-7. Common Element Two-Phase Clock

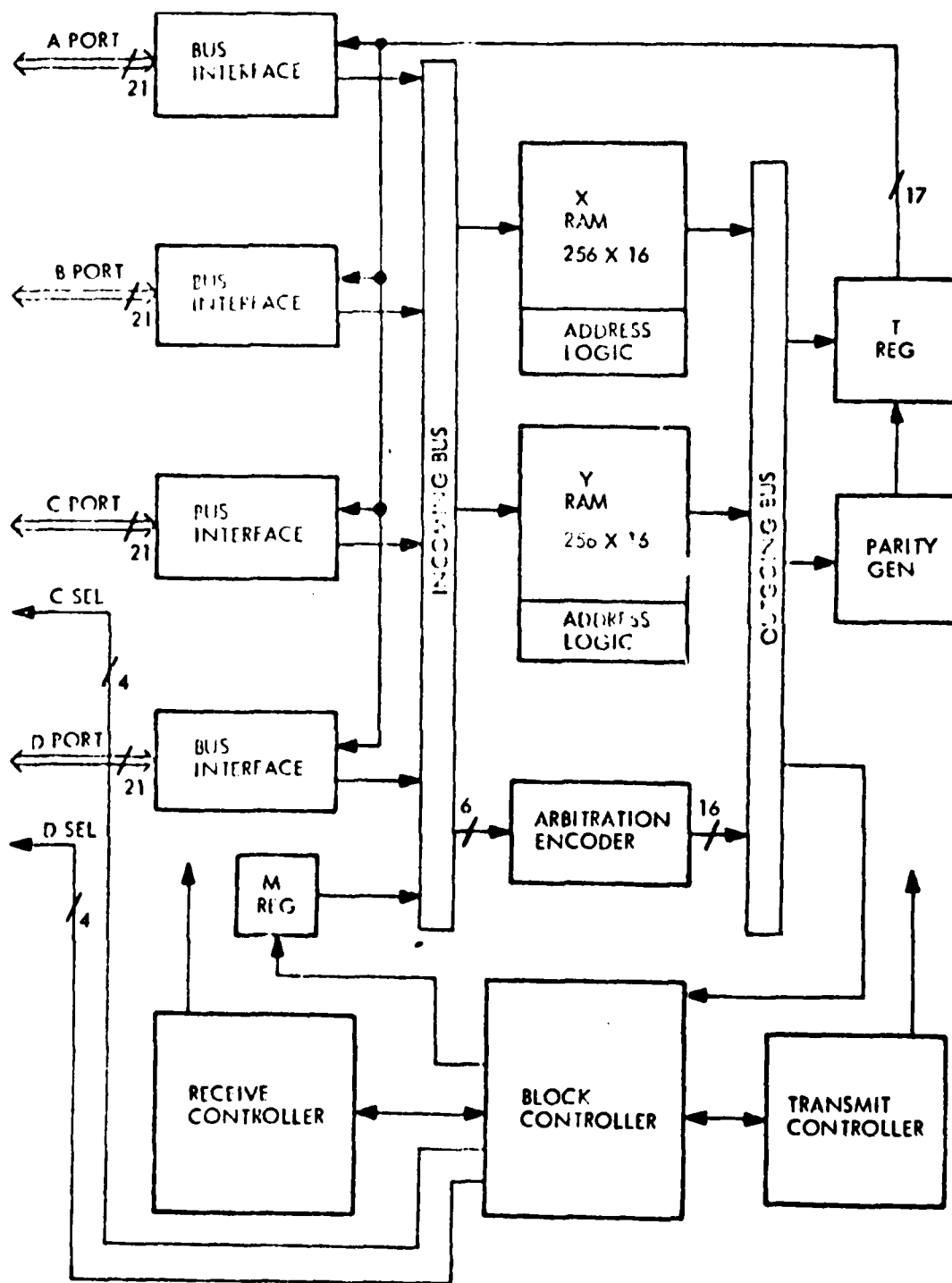


Figure 4-8. IOC Module, Block Diagram

and the data block is sent to its destination through another port by the Transmit Controller.

The IOC implements intercluster and system I/O block transfer through redundant channels so that failures of single IOCs or entire busses can be tolerated without loss of performance. The A and B ports are always connected to the corresponding busses in the cluster where the IOC is resident, thereby providing dual paths to all other elements in that cluster. In multicluster systems, the C and D ports of an IOC in an executive cluster connect to A and/or B busses of two different slave clusters. In this way, only $N + 1$ IOCs in the executive cluster are needed for reconfiguration in the event of failures of an IOC or of either of the A or B busses in N slave clusters. The IOC's C and D ports communicate with peripherals in a direct mode; two four-bit select-code outputs (See Figure 4-8) are used to select one of up to 16 peripheral devices. The Input Synchronizer and Output Synchronizer are used as peripherals in FTWRP; each is connected to the C-port of its IOC.

The Bus Interfaces include bus transceivers and input registers, as well as header decoding, bus-access arbitration, and parity check logic as discussed in Section 4.1.3.1.2. The four interfaces are essentially identical, except that the C and D ports interpret certain message codes which could only have come from an executive CE as invalid; executive messages are only allowed to enter IOCs through the A or B ports. The independent receive and transmit controllers, implemented in high-speed logic, carry out the arbitration, destination decoding, data transfer and parity checking by interacting with the bus interfaces and control lines. Fast RAMs are used as dual buffers for simultaneous transmit and receive through separate ports (or through the same port for test purposes). The IOC responds to "status request" messages from the executive CE by formatting and sending a "status return" message containing at least the following: IOC virtual address, current mode of operation, selected port, and current bus selected. Error conditions, such as invalid message codes received on C or D ports, are also included in this message.

The multiport architecture of the IOC uses a variety of tests on a multitude of data paths and circuit elements in addition to the pervasive parity checking to assess performance. Each port is directed to itself as a destination through the various possible bus paths; for example, through the C port of an executive cluster into a slave cluster's A bus, into an A port of some spare device resident there, out through that device's B port, into the D port of another executive-cluster IOC, out through its A port and back to the first IOC's A port. This checking, under control of the executive, exercises and tests spare data paths in the system. The controller in a spare IOC is tested by commanding it to perform functional tests such as accepting a message, modifying the contents in some unique way and then returning the result to the executive for verification.

The IOC block controller, block-diagrammed in Figure 4-9 contains the microprocessor controller for generating the control signals for the high-speed receiver controller and transmit controller. The block controller contains a Signetics 8x300 bipolar Schottky microprocessor which executes 16 bit instructions in 250 nanoseconds.

Data handling and I/O device addressing are accomplished via the 8-bit interface vector (IV) bus. The IV bus is supported by four control lines and the 8X300-generated clock. The block controller contains fourteen Signetics 8T32's which are 8-bit latched addressable bidirectional I/O ports. The microprocessor addresses the port. If the address matches the 8T32's internally-programmed address, the port is enabled, allowing data transfer. With these ports, 16-bit words are read from the IOC outgoing bus, and 16-bit words transferred to the incoming bus. Transmit interrupt, transmit status, receive interrupt, and receiver status bits are sampled. Bus selects, RAM clocks, register clocks, and device select signals are generated at the outputs of the IV registers.

The IOC firmware is discussed in detail in Section 6.2.

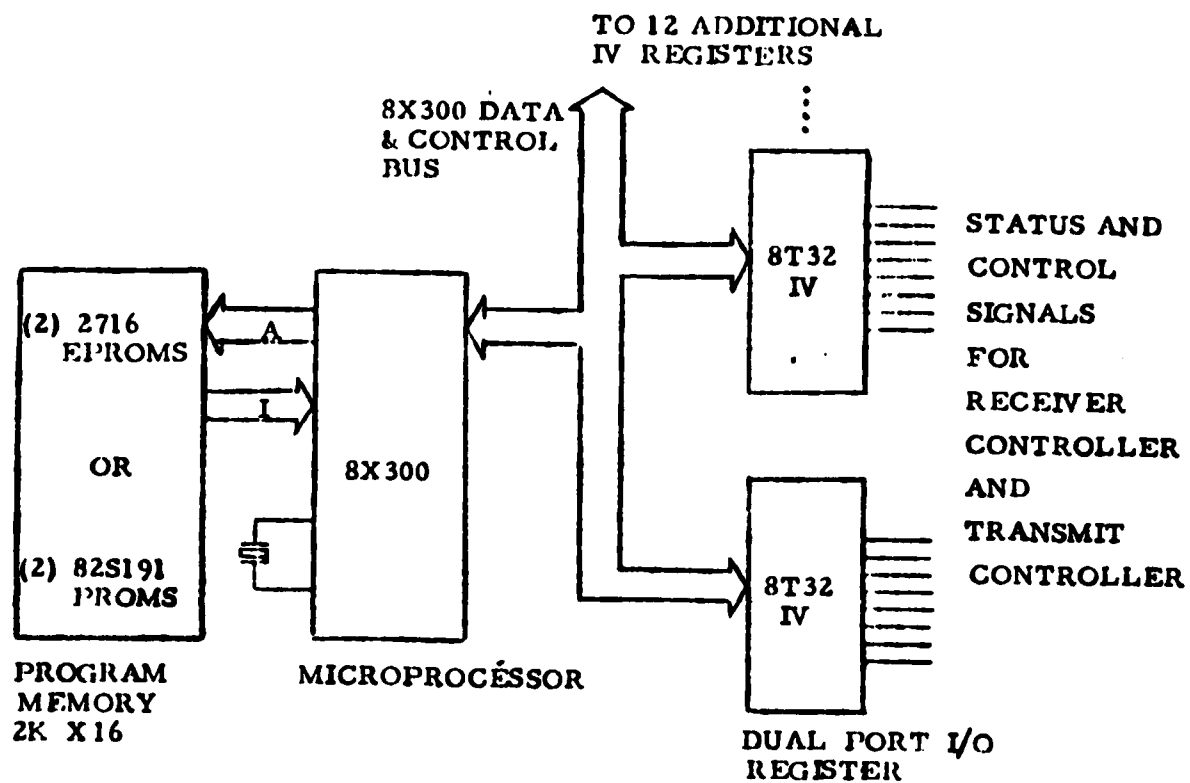


Figure 4-9. IOC Controller Block Diagram

4.3 Terminal Interface Element

The Terminal Interface Element (TIE) is designed to interface an intelligent terminal to the two 16-bit open collector data buses in the fault-tolerant weather radar system. Use of this interface permits high-speed asynchronous data transfers independent of the relatively slow, micro-processor-based terminal. The TIE communicates with the intelligent terminal via a 24-bit parallel I/O port, and with other elements via an extension of the dual 16-bit data bus. The TIE is mounted on the back of the terminal along with its own 5-volt power supply.

A block diagram of the TIE is shown in Figure 4-10. It consists of a Common Element Bus Transceiver with some extra hardware to permit communication with the terminal. The majority of this extra logic is devoted to interfacing the 16-bit buses of the I/O transceiver to the 8-bit bi-directional bus of the terminal. This is accomplished by placing four 8-bit tri-state registers on the 8-bit bus which alternately clock the most-significant part (MSP) and the least-significant part (LSP). Control of which part is loaded at each clock is performed by the LSP/MSP flip-flop, which toggles each time a read or write is ended.

The TIE's transmit capability may be disabled by raising the ITENBL line. This permits the TIE to listen on the bus and recognize messages without responding with acknowledges. The purpose of this feature is to allow the TIE to "eavesdrop" on the bus by taking another elements' virtual address and listening to all messages sent to it. The eavesdrop capability has never been tested, but should work in principle.

The overall operation of the TIE is best shown by describing each of the inputs and control lines in the system. The outputs are discussed in Section 4.3.1 and inputs in Section 4.3.2. Section 4.3.3 describes the status byte of the TIE, and the possible transmitter and receiver states.

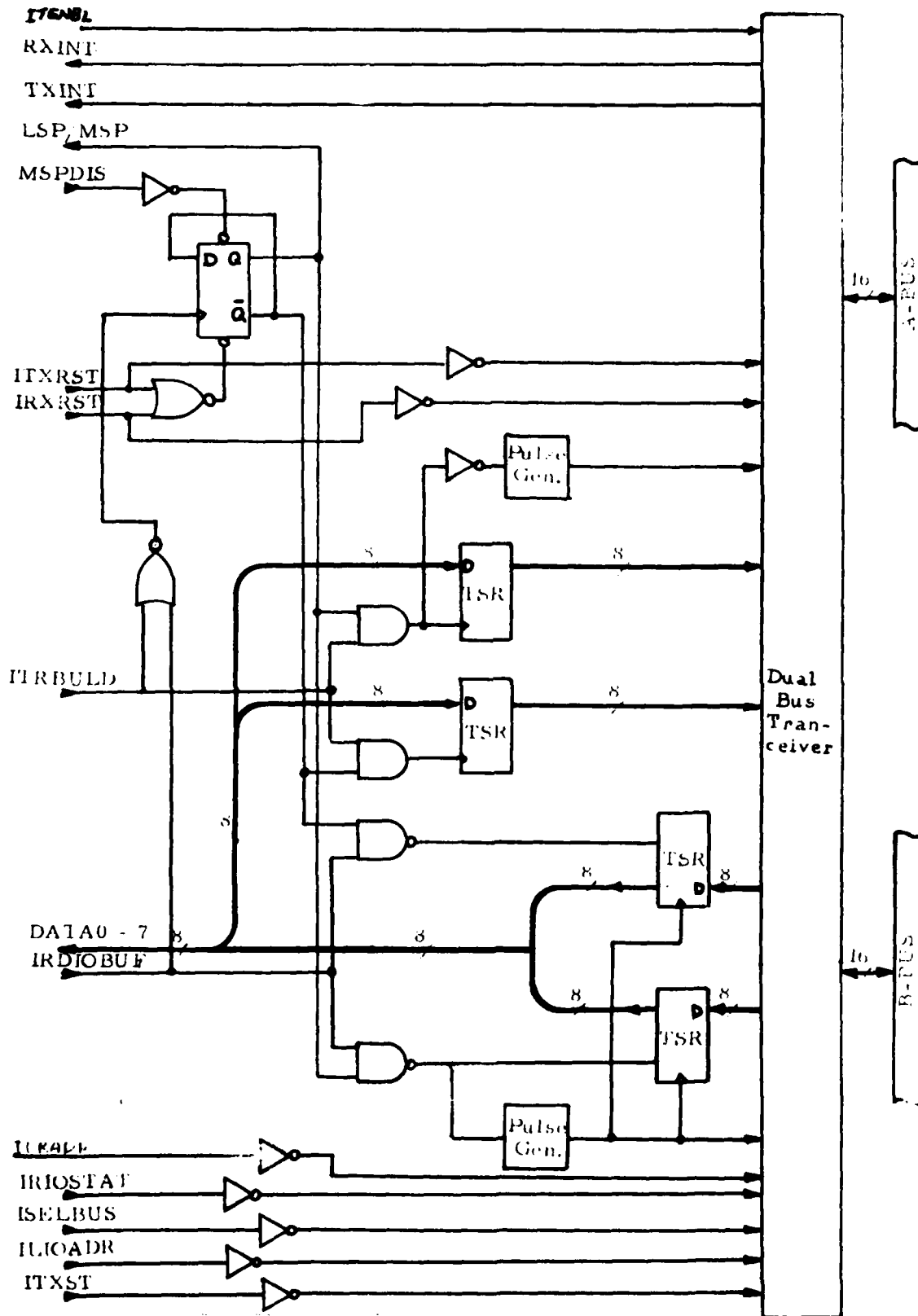


Figure 4-10. Terminal Interface Element Block Diagram

4.3.1 Outputs from TIE

4.3.1.1 LSP/ $\overline{\text{MSP}}$

- o When high, indicates next byte transferred is least-significant part of 16-bit word.
- o Must be low at the beginning of each read or write between terminal and TIE.
- o Reset by ITXRST and/or IRXRST.

4.3.1.2 RXINT

- o Signifies that the I/O receiver requires service.
- o Generated upon completion of a received block or when a receiver fault is recognized.
- o Reset by IRXRST.

4.3.1.3 TXINT

- o Signifies that the I/O transmitter requires service.
- o Generated either upon completion of transmission or when a transmitter fault is recognized.
- o Reset by ITXRST.

4.3.2 Inputs to TIE

4.3.2.1 IRIOBUF

- o Outputs 8 bits of receiver RAM to data bus when high.
- o LSP/ $\overline{\text{MSP}}$ determines which 8 bits are output.
- o When LSP/ $\overline{\text{MSP}}$ is high, receiver RAM address advances on position.
- o Must be left low when receiver is idle.

4.3.2.2 IRIOSTAT

- o Outputs transceiver status to data bus when high.

4.3.2.3 ISELBUS

- o Selects transmit bus according to state of DATA0 (High = B, Low = A).

- o Must stay low when transmitter is triggered.
- o Clocks on negative edge.

4.3.2.4 ITXRST

- o Resets transmitter to idle from any state when high.
- o Resets LSP/ $\overline{\text{MSP}}$.
- o Clears TXINT.

4.3.2.5 IRXRST

- o Resets receiver to idle from any state when high.
- o Resets LSP/ $\overline{\text{MSP}}$.
- o Clears RXINT.
- o Must stay low when receiver is idle.

4.3.2.6 ILIOADR

- o Loads new virtual address from data bus on negative edge.
- o New address will be used immediately by receiver.
- o Transmitter will use new address on first ITXST after loading.
- o Must stay low while transmitter is triggered.
- o Does not affect the data block header word.

4.3.2.7 ITXST

- o Starts transmit sequence when high.
- o Uses latest loaded virtual address.
- o May be used to reattempt transmission of a block previously loaded into transmitter RAM if no ITXRST has been given.

4.3.2.8 ITRBULD

- o Loads one 8-bit byte from data bus into transmit register when high.
- o LSP/ $\overline{\text{MSP}}$ determines which byte is loaded.
- o When LSP/ $\overline{\text{MSP}}$ is high, negative edge of ITRBULD clocks data into transmit RAM.

- o Load only when transmitter is idle.

4.3.2.9 MSPDIS

- o Disables loading or reading of most-significant byte from data bus.
- o LSP/ $\overline{\text{MSP}}$ is set high.
- o LSP/ $\overline{\text{MSP}}$ does not toggle when ITRBULD or IRDIOBUF are high.

4.3.2.10 ILRADR

- o Loads contents of data bus into receiver RAM address counter.

4.3.3 Transceiver Status Word

The transceiver outputs an 8-bit status word to the data bus whenever IRIOSTAT is raised. Included in this word are the transmitter state, receiver state, and last transmit bus. Tables 4-5 through 4-7 list the various possible states. Bit 0 gives the transmit bus, where a logic 0 denotes bus A, and a 1 denotes bus B. Bits 4 through 1 contain the transmit status, and bits 7 through 5 contain the receiver status. The various states of the receiver and transmitter are described in the next sections.

4.3.3.1 Receiver States

4.3.3.1.1 Idle

- o No data waiting in RAM.
- o May be receiving.
- o Do not reset receiver when it is in this state.

4.3.3.1.2 Parity Error A

- o Parity error has occurred during reception on bus A.
- o Sender is still on bus.

4.3.3.1.3 Parity Error B

- o Parity error has occurred during reception on bus B.
- o Sender is still on bus.

Table 4-5. Receiver State

<u>DATA7</u>	<u>DATA6</u>	<u>DATA5</u>	<u>STATE</u>
H	L	L	Idle
H	H	L	Parity Error A
H	L	H	Parity Error B
L	H	L	Fault A*
L	L	H	Fault B*
L	L	L	Full*

No other states of DATA5 through DATA7 occur.

*(interrupt)

Table 4-6. Last Transmit Bus

<u>DATA0</u>	<u>BUS</u>
L	A
H	B

Table 4-7. Transmitter State

<u>DATA4</u>	<u>DATA3</u>	<u>DATA2</u>	<u>DATA1</u>	<u>STATE</u>
H	H	H	H	Triggered
H	H	L	H	Bus Busy*
H	L	H	H	ARB Fault*
H	L	L	H	Reply Fault*
L	H	H	H	REC Busy*
L	H	L	H	Parity Fault*
L	L	H	H	Time Fault*
L	L	L	H	Done*
L	L	L	L	Idle

No other states of DATA1 through DATA4 occur.

*(interrupt)

4.3.3.1.4 Fault A

- o Incomplete block received on A.
- o Parity error or time elapsed.

4.3.3.1.5 Fault B

- o Incomplete block received on B.
- o Parity error or time elapsed.

4.3.3.1.6 Full

- o Data block received without error.
- o Read onto data bus any time before next IRXRST.

4.3.3.2 Transmitter States

4.3.3.2.1 Triggered

- o Sending or waiting to send.
- o Will proceed to an interrupt state.
- o ITXST has been received.
- o Maybe reset from this state directly.

4.3.3.2.2 Bus Busy

- o Unable to get bus and win arbitration in required time.

4.3.3.2.3 Arbitration Fault

- o Too much time taken in arbitration.

4.3.3.2.4 Reply Fault

- o No reply from receiver.
- o May mean parity error on header.

4.3.3.2.5 Receiver Busy

- o Receiver answered busy.

4.3.3.2.6 Parity Error

- o Transmission incomplete due to parity error.

4.3.3.2.7 Time Fault

- o Transmission incomplete
- o Transmission took too much time.
- o Receiver did reply to header.

4.3.3.2.8 Done

- o Transmission completed.
- o Must get IRXRST to go to "idle".

4.3.3.2.9 Idle

- o Must get ITXRST to get to this state.
- o Load transmitter buffer only when transmitter is in this state.
- o Counts block length: number of words written between ITXRST and ITXST.
- o Goes to "triggered" on receipt of ITXST.

4.4 Input Synchronizer

The Input Synchronizer (IS), which resides in slot T13 of the Pulse Pair Processor (PPP), receives digital video data from the Pulse Pair Processor A/D converters and sends it in block format to the input IOC for distribution to the CE's. Processing in the FTWRP is distributed according to range, as shown in Figure 4-11, where the total number of processing CEs (n) is currently 5.

Since reflectivity and coherent-channel information must be processed separately in each CE, they are treated separately by the IS; therefore, for each range interval, two blocks of data (In-phase and Quadrature phase/amplitude information, and reflectivity) are transmitted. The format for the block is shown in Figure 4-12. The first word contains the current radar pulse number (needed for processing CE synchronization), and block number. The second word contains the following PPP control settings: radar pulse width T_p or clock speed (2 bits) and number of range cells, N_{RC} , (2 bits). The remaining words contain coherent channel I and Q data (16 bits) or reflectivity data (8 bits). The N_{RC} and T_p parameters are defined in Table 4-8.

The IOC resets the IS by activating Select Line 2 (SEL2). When SEL2 goes inactive, the IS waits for the next radar trigger pulse from the PPP before accepting data from the A/Ds. The IOC can request either I & Q data or reflectivity data by activating SEL0 or SEL1, respectively.

The timing diagram of Figure 4-13 shows an actual case of how reflectivity and I and Q blocks are handled. As Figure 4-13 indicates, the IOC will input I and Q data, or "pulse pair" data (P) for the first range interval, but allow reflectivity (power) data (Z) to stay in the synchronizer until the fifth range interval. Thereafter, the power and pulse data are offset by four range intervals. This built-in offset prevents bottlenecks at the CEs, and permits processing to proceed more smoothly.

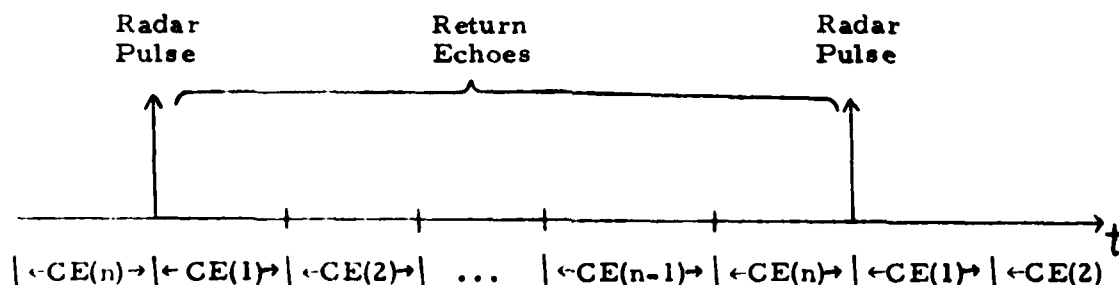


Figure 4-11. Distribution of Processing in FTWRP

WORD

0	0	BLOCK NUMBER	PULSE NUMBER
	MSB	MSB	
1	0	SPARE	T _P N _{RC}
2	I	Q	
3	I	Q	

⋮

0	0	BLOCK NUMBER	PULSE NUMBER
	MSB		
1	0	SPARE	T _P N _{RC}
2	0	POWER	
3	0	POWER	

⋮

Figure 4-12. Input Synchronizer Block Format

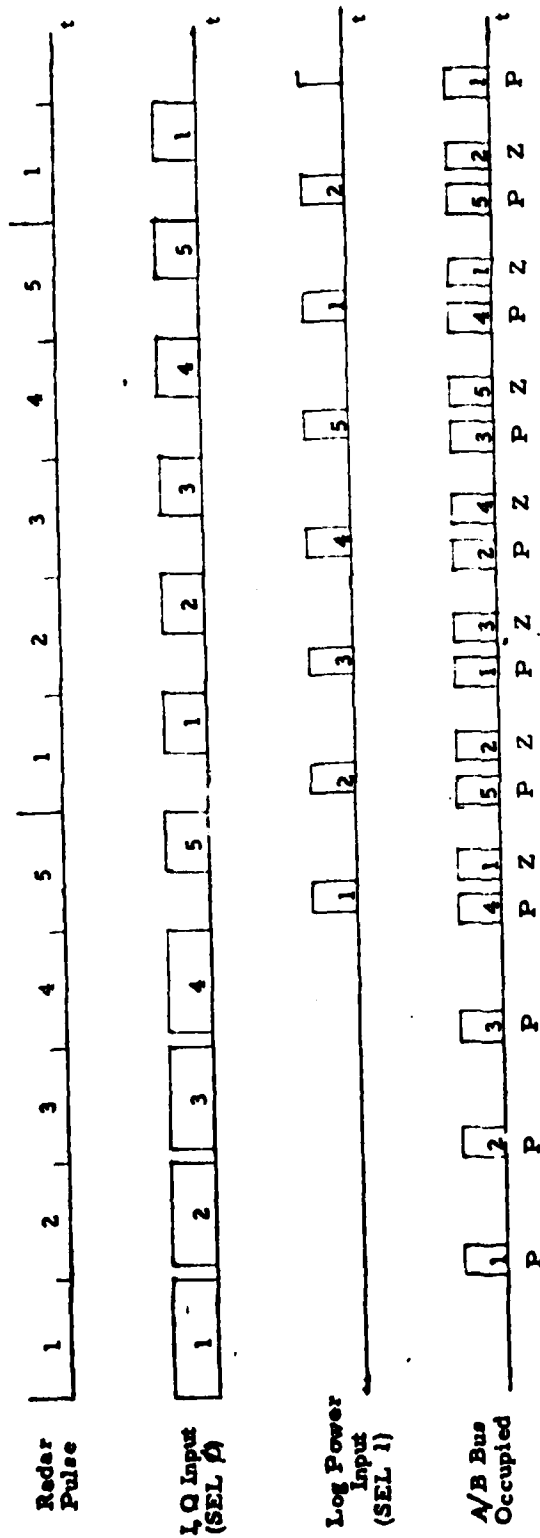


Figure 4-13. Continuous Pulse Scheme Timing.
Z Denotes Reflectivity Data, P is Coherent Data

Table 4-8. N_{RC} and T_p Definitions

N_{RC}		T_p	
00	256	00	0.5 μs
01	512	01	1.0 μs
10	768	10	2.0 μs
11	1024		

4.4.1 Input Synchronizer Hardware Description

A block diagram of the input synchronizer is shown in Figure 4-14. The synchronizer accepts data from the PPP and stores it in RAM. Separate RAMs are used for I and Q data and for Log Power data, since the Log Power buffer needs to store up to the entire 1024 cells, whereas the I and Q buffer needs to store only one-fifth as many cells. When a block is requested by an IOC, after the two radar parameter words are sent, data is read from the RAM and put onto the bus. An input address counter and an output address counter assure that data is stored into and read from sequential locations. A priority control circuit gives priority to the write operation, so that all data is received from the PPP. If the synchronizer is sending data to the IOC, and a priority write occurs, it will finish sending the current word, perform the write, and then continue sending to the IOC.

After the last address in RAM is used, it will start writing at the first address, which should have already been sent to the IOC. This memory arrangement is similar to a FIFO. In the event that the IOC is too slow, an address comparator warns the IOC, by dropping the parity line, that data has been written into an address that has not yet been sent to an IOC. The address comparator is implemented by an up/down counter which counts up for every word read from the PPP and down for every word sent to an IOC. An overwrite condition occurs when the counter produces a carry from the MSB. The address comparator also prevents the IOC from reading a word twice. This condition occurs when all the up/down counter outputs are '0'. If the IOC has read all the current words in RAM, it must wait until a write occurs before it can read again.

Two Block Counters, one for I and Q data, and one for Log Power data, are used to count the number of blocks sent to the IOC. These counters are cleared with every radar pulse. A MOD 256 counter continuously counts radar pulses. This counter is cleared only at power on, and when requested by the IOC. Radar parameter information, T_p and N_{RC} , is stored in a register.

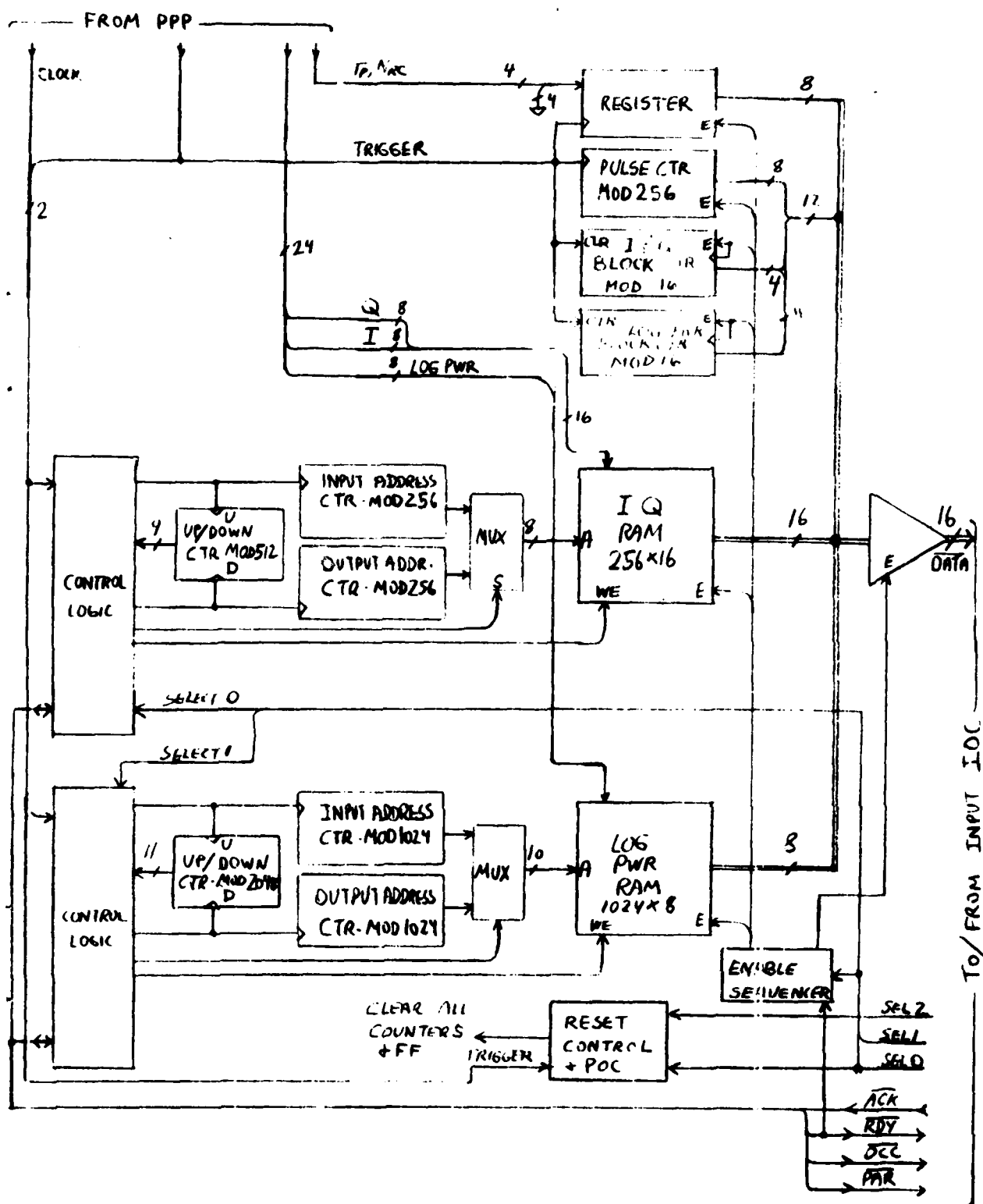


Figure 4-14 Input Synchronizer Block Diagram

The enable sequencer enables the correct radar parameter words and data words onto the bus at the proper time. When I and Q data is requested by the IOC, with a SEL0, the I and Q block counter is enabled along with the pulse count for the first word. The register, containing radar parameters, is enabled next, and then the I and Q data words are sent. The format is the same for Log Power data except the Log Power block counter is used.

4.4.2 Data Transfer Timing

Figure 4-15 shows timing for a typical block transfer in the input synchronizer. After the radar trigger pulse, two range cells of data are clocked into the buffer RAM before the IOC requests data by activating the select line. The IS immediately drops the Bus Occupied line (\overline{OCC}), and Data Ready (\overline{RDY}), signalling to the IOC that a word (the first header word) is ready for transfer. However, by the time the Acknowledge (\overline{ACK}) comes back from the IOC, a new data word is ready from the PPP. Therefore, \overline{RDY} is held high while the data is written into the RAM.

Once the write operation is complete, the RAM address is switched back to the output mode and \overline{RDY} is lowered once more to output the second header word. Then, the RAM is read and output consecutively until it is emptied (3 words are output) before the next radar sample is available. From this point on, \overline{RDY} is pulsed only when a new sample comes in. The transfer stops when the IOC raises the select line to the inactive state. The next time the same select line is activated, the IS will output the two header words again, then start outputting data from the RAM address where it left off before.

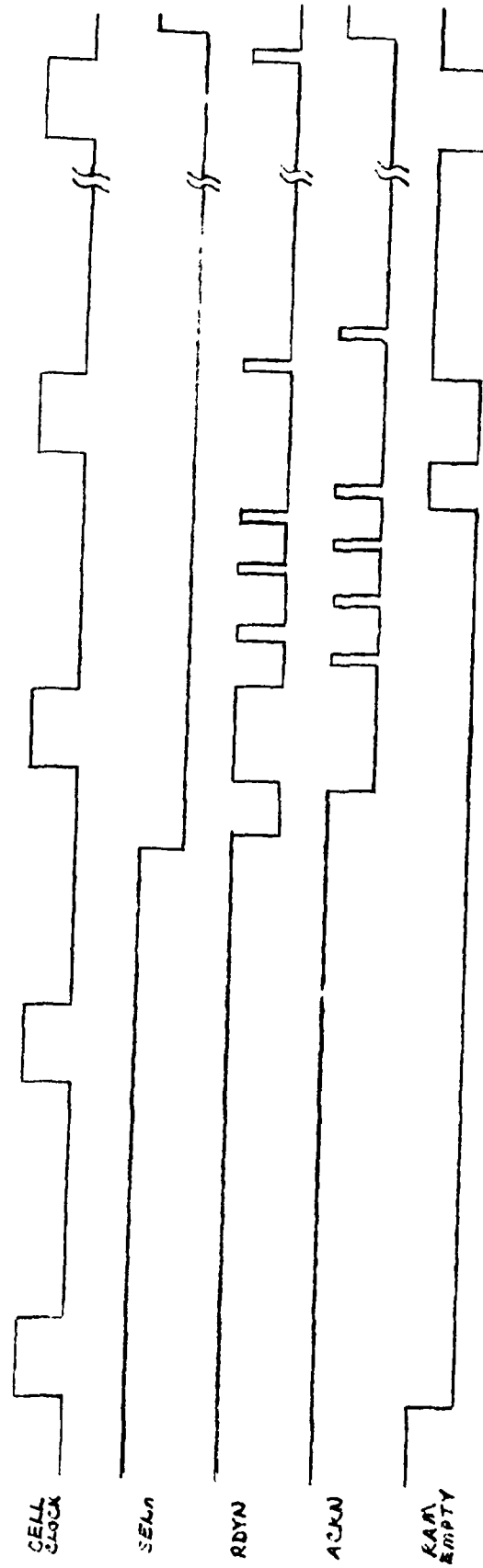


Figure 4-15. Input Synchronizer Timing

4.5 Output Synchronizer

The Output Synchronizer, connected within the FTWRP system as discussed in Section 2.4, accepts processed radar data from the C port of the output IOC then performs code conversion and data buffering operations to provide an output compatible with the PPP Recorder Encoder. As each CE completes its processing, it formats an output buffer containing data having the format shown in Figure 4-16. The message is made up of multiple blocks (packets) except for the case $N_{RC} = 256$ in which all data fits in one block since

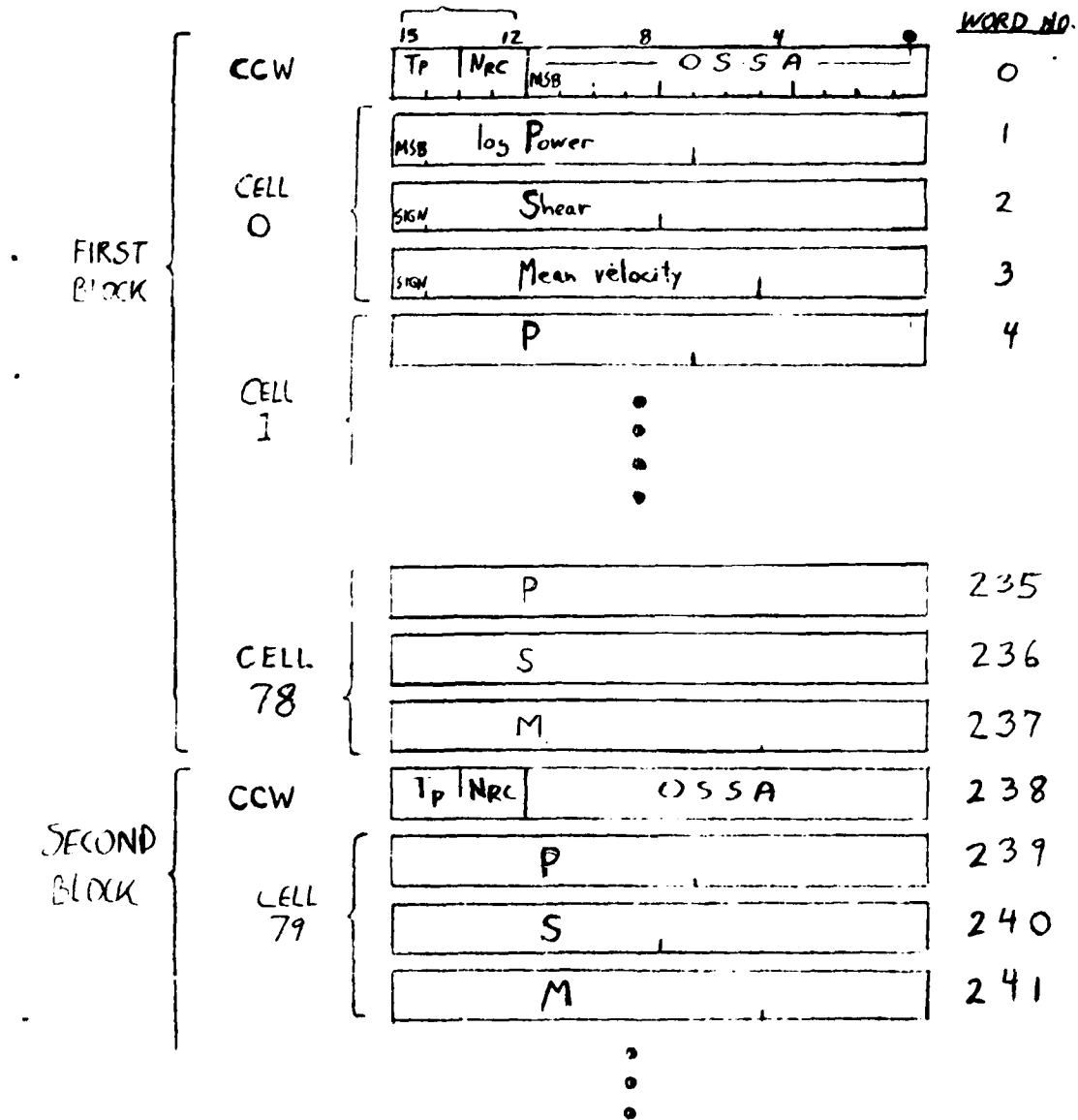
$$\frac{256 \text{ total cells} \times 3 \text{ variables}}{5 \text{ CEs}} = 154 < 240 \text{ words/block.}$$

The buffer area corresponding to each block begins with a CCW (Channel Control Word) which specifies T_p (radar pulse width), N_{RC} (number of range cells), and OSSA (Output Synchronizer Starting Address). OSSA defines the desired location in the Output Synchronizer's buffer of the first data word after the CCW of this block. The CCW are transparent both to DOS-0 and the output IOC, which operates in its dynamic output mode and provides the message blocks with headers and other control words. The first word of each block which the output synchronizer needs is the CCW; the header and three following words are ignored.

4.5.1 Output Synchronizer Hardware Description

The output synchronizer (OS) block diagram appears in Figure 4-17 where the interface with the IOC is shown at the left and connections to the PPP Recorder Encoder appear at the right. The heart of the OS is the buffer where packets of the form shown in Figure 4-16 are assembled into the 28-bit-word by 256, 512, 768, or 1024-range-cell format needed by the Encoder. As power, Shear, and Mean-Velocity words are received at a rate controlled by the IOC through the \overline{RDY} line, they are either converted to sign-magnitude by a PROM or not before being loaded into separate power, Shear and Mean registers. These data registers have tri-state outputs for compatibility with the buffer RAM's bidirectional I/O pins. Which words are to be converted is defined by jumper programming in the OS, since compatibility in the existing tapes having sign-magnitude mean velocity may be desirable,

Also see Table 4-8



OSSA = Output Synchronizer Starting Address

Figure 4-16. Format of CE Output Buffer for $N_{RC} = 512$ or 768 (2 blocks) or $N_{RC} = 1024$ (3 blocks). For $N_{RC} = 256$, only one block is needed.

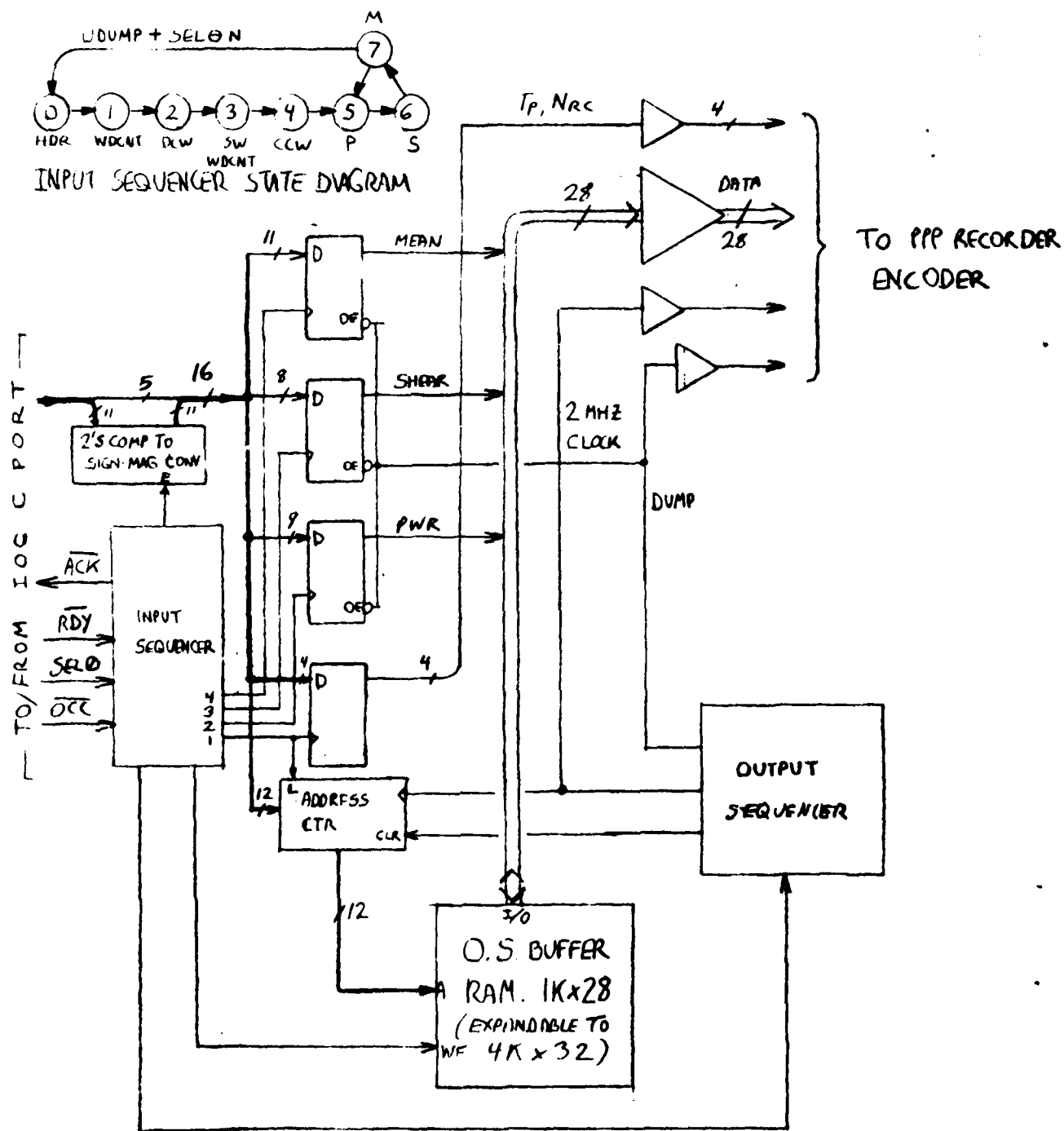


Figure 4-17 Output Synchronizer Block Diagram.

whereas there is no precedent for shear which will be recorded in place of variance, an unsigned quantity. The jumpers are presently set to convert shear and Mean, but not power.

Each time the three registers are loaded, a write strobe to the OS buffer RAM stores the resulting 28-bit word, then the address is incremented. Although the 1K x 28 buffer is matched to the current capacity of the PPP Recorder Encoder, extra address bits are provided so that the capacity can be extended to 2K, 3K or 4K by adding more 2114 1K-by-4 static MOS RAM chips. More bits per word can also be similarly added. The circuit panel is wired to accept more 2114s so that the buffer could be expanded to 4K x 32.

Addresses for the buffer are generated in a 12-bit counter both for input, where the counter is initialized to O SSA prior to each packet transfer, and for output where it is reset to zero and counts up to the appropriate N_{RC} -- 256, 512, 768 or 1024. If the encoder is modified to accept more cells, some changes in the circuitry which decodes N_{RC} and stops the output "dump" sequence and the counter will be needed; however, the counter itself can handle up to 4K cells.

Operation of the OS is controlled by separate input and output sequencers (see Figure 4-17). The input sequencer timing is related to that of the IOC, whereas the output sequencer contains a 16 MHz crystal controlled oscillator. Details of overall OS timing, in which the two sequencers run mutually-exclusively in time, are presented in the next subsection.

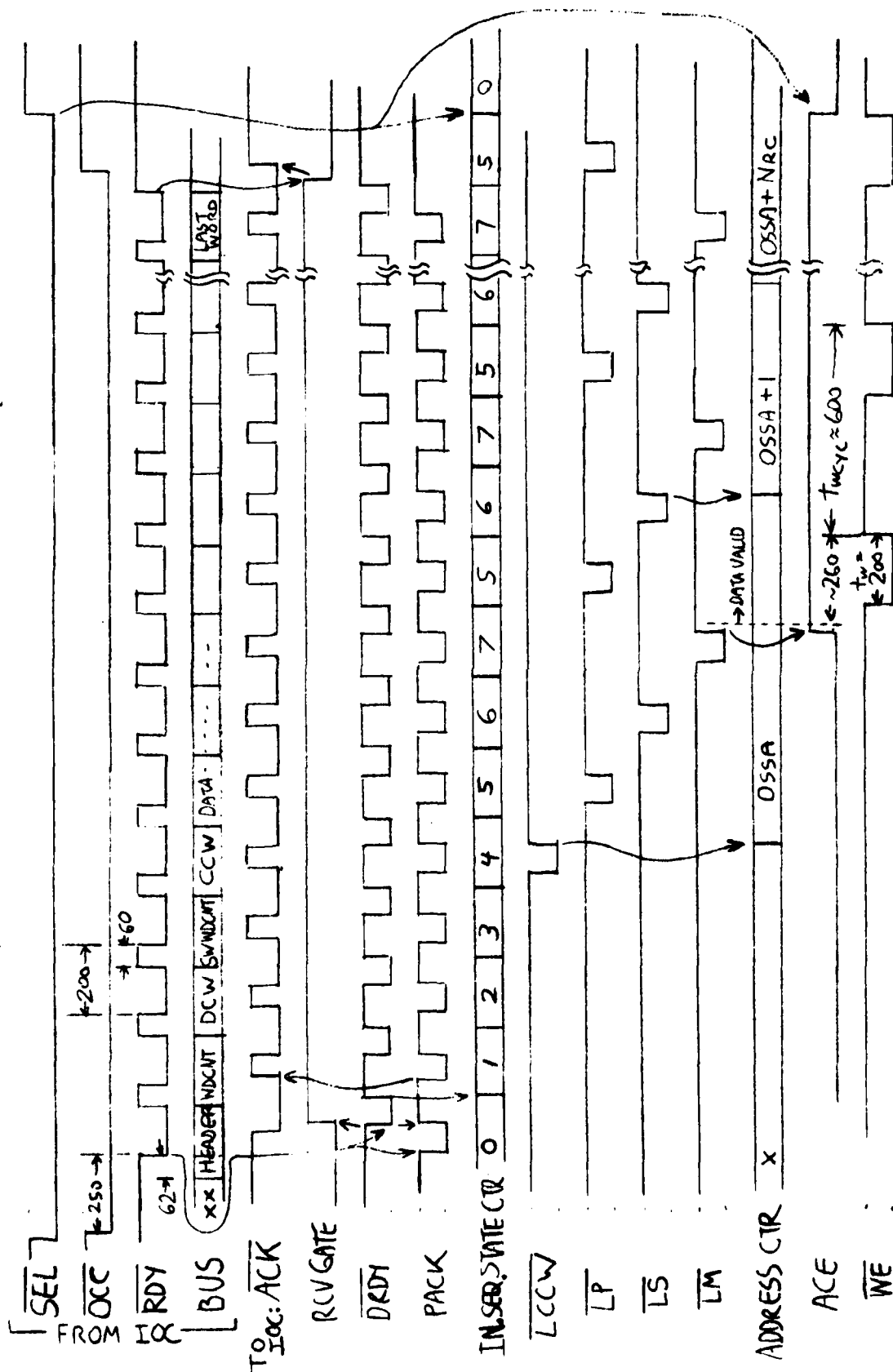
The OS is constructed on an AUGAT 8136-UG6-27 universal wirewrap panel having 27 columns on 0.3" centers and 50 rows on 0.1" centers. Logic is a combination of Schottky and Low-power Schottky MSI TTL, while the buffer memory is implemented in static NMOS. The OS is housed in a 19" x 3.5" rack-mounting unit which must be located near and powered by the Encoder -- about 2.6 Amperes at 5.0 volts is needed. Applicable drawings are listed in Appendix G.

4.5.2 Data Transfer Timing

Since each CE is performing the same processing on approximately the same number of range cells, they will all have outputs ready at about the same time. The Output Sequencer will wait about 20 milliseconds after the last message, to ensure that all have been received, then will begin an output sequence to "dump" the OS buffer contents into the PPP Recorder Encoder. During the dump, the input sequencer is disabled and would not respond to attempts at communication by the IOC.

Timing of the Input Sequencer is illustrated in Figure 4-18. The top four waveforms represent the four control lines from the output IOC's C-port, while ACK is a handshake signal returned to the IOC by the Input Sequencer. Cause-effect relationships are indicated in the Figure by arrows. When a message block is about to be output from the IOC C-port, its \overline{SELO} line goes low then its \overline{OCC} control line becomes low. These events are interpreted by the Input Sequencer as indicating the beginning of a block. A 16-bit counter keeps track of the initial sequence of words within each packet, then cycles through three of its states as each group of Power, Shear, and Mean words is transferred (see state diagram in Figure 4-17). The Input Sequencer loads the first word, the CCW, partly into a four-bit register to contain T_p and N_{RC} and partly into a 12-bit counter which addresses the OS Buffer. Load strobes for the CCW and the three data registers are decoded from the state counter states and ANDed with the PACK signal. The timing diagram also indicates buffer write timing and shows that considerable margins exist for the Motorola MCM21L14-20, which has a minimum write cycle time of 200 nsec, write time of 120 nsec, and data-to-write overlap of 120 nsec.

Output sequencer timing is much simpler and requires no diagram for explanation. When a "dump" is initiated, the Address counter is reset to zero, then incremented at a two-MHz rate while the memory is sequentially read and its contents duplicated in the shift register array of the Encoder. The "dump" is terminated and the address counter is stopped when the OC STOP signal indicates that N_{RC} cells worth of data have been transferred.



NOTE: TIMES IN N-SEC FOR 5MHZ BUS RATE (ACTUAL RATE ≈ 4 MHZ)

FIGURE 4-18 OUTPUT SYNCHRONIZER INPUT SEQUENCER AND BUFFER WRITE TIMING.

The 2 MHz clock, derived from a 16 MHz crystal-controlled oscillator, is buffered and sent to the Encoder along with the 28 bits of data, a buffered DUMP gate, and four lines for T_p and N_{RC} .

REFERENCES

- 4-1. The Bipolar Microcomputer Components Data Book, Second Edition,
Texas Instruments, 1979.
- 4-2. K.A. Smith memo, "FTSP Programmer's Handbook Part I and II",
dated 31 August 1979, KAS:79:10, EM79-0521.

5. DETAILED SOFTWARE DESCRIPTION

5.1 Distributed Operating System Level 0 (DOS-0)

DOS-0 is primarily an interrupt-driven real-time executive responsible for the management of resources of an individual common element. Its fundamental requirements lie in the area of message processing and as such, it acts as a message switching mechanism for all input/output (I/O) with the cluster on which resides the Common Element. DOS-0 also assumes the responsibilities of CE initialization, task (user) request processing, fault monitoring, and interrupt-handling.

The structure of the DOS-0 hierarchy (Figure 5-1) reflects the interrupt structure of the common element architecture, and the individual components represent an independent collection of processing modules. In the following sections we present more detail on the functions of each component.

5.1.1 Initialization

Upon power-up or system restart, the initialization is invoked to provide an orderly environment in which the user tasks may execute. This routine provides for the following activities:

- a. Memory self-test
- b. Initialization of level 0 system data base elements
- c. Execute a wait to permit level 1 communications
- d. Request load of level 1 task code
- e. Start level 1, if loaded properly
- f. Enter an idle or wait loop, awaiting assignment by level 1 system.

In FTWRP, since the level 1 operating system resides in the Intelligent Terminal, the request to load the level 1 task will always fail. DOS-0 will then simply enter its idle loop, waiting for communication from the executive.

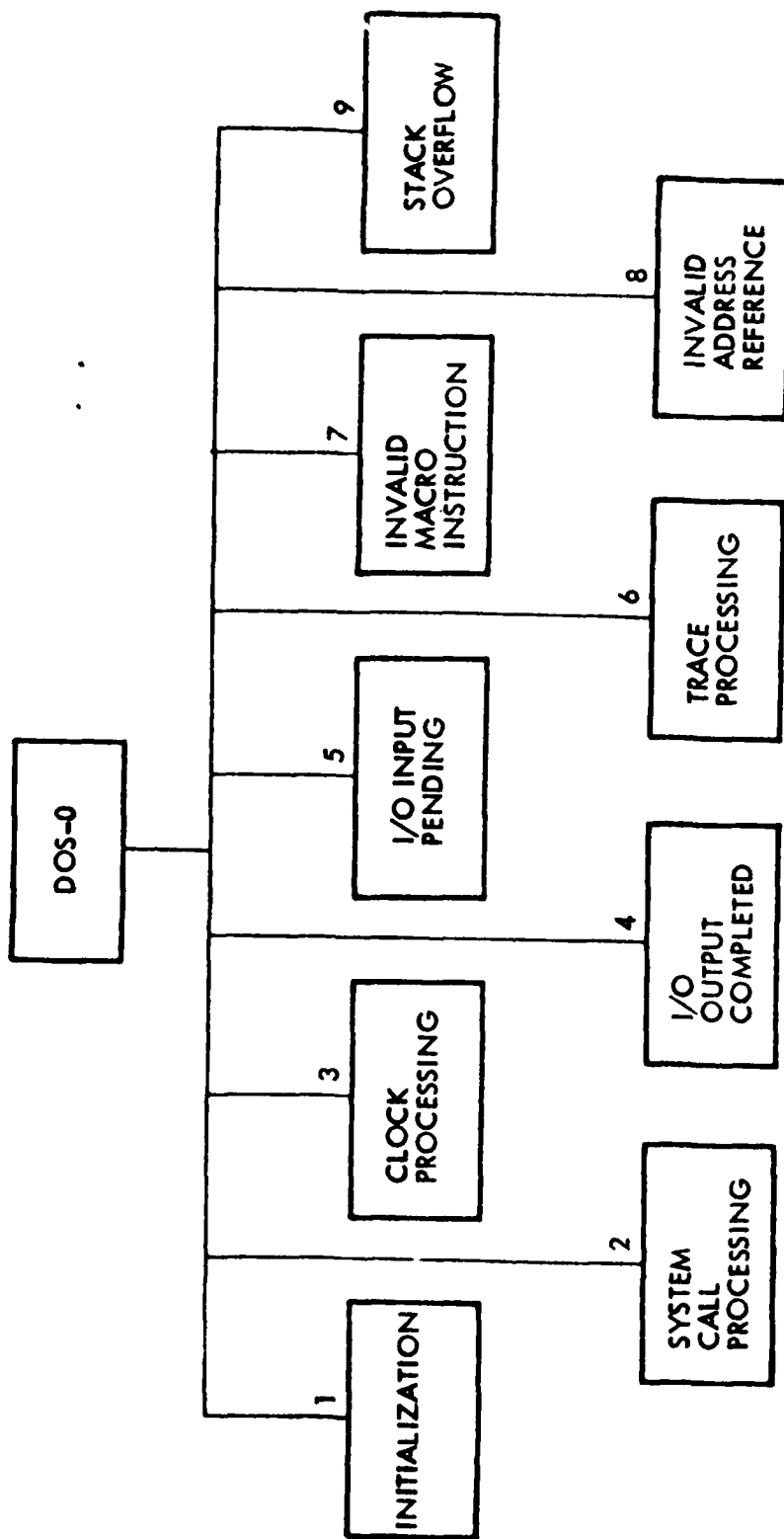


Figure 5-1. DOS-0 Hierarchy

5.1.2 Task Environment

The Fault-Tolerant Signal Processor software structure is based on the concept of a task which is considered as that program currently executing on a given CE or I/O port. Tasks are identified to the system by an identification number or task address (virtual address) which is independent of the physical address of the card on which the task is executing. Thus one physical CE may assume, in time, any sequence of virtual addresses depending on the system load and DOS-1 task scheduling algorithms. DOS-0 has the capability to change the virtual address of the CE in which it resides, but only at the direct or indirect authorization from DOS-1.

The following sections describe the task environment of DOS-0 in detail, while Appendix B presents a tabular form of the data structures employed.

5.1.2.1 Memory Management

DOS-0 is resident in PROM located at the upper 4K of the CE's main memory address space (See Figure 5-2). It also utilizes a small block of the 16K onboard RAM for current task information, temporary storage, and a system stack.

User task code is loaded upon request through DOS-0 from a Common Memory, which, in FTWRP, is provided via the intelligent terminal. Task code is divided into two distinct sections, the pure, executable program code, and pure data space. Data records are relocated to the upper portion of the physical RAM address space by DOS-0 at load time. The user stack is then set up to utilize the RAM area between the user code and data space.

5.1.2.2 User Task Prologue

The first 256 words of the RAM address space are reserved for the task prologue. The prologue supplies information about the task to DOS-0, such as starting addresses for various entry points, task ID number, data base structures and size, etc. It must be explicitly assembled into every program at assembly time. The format for the prologue is shown in Table 5-1.

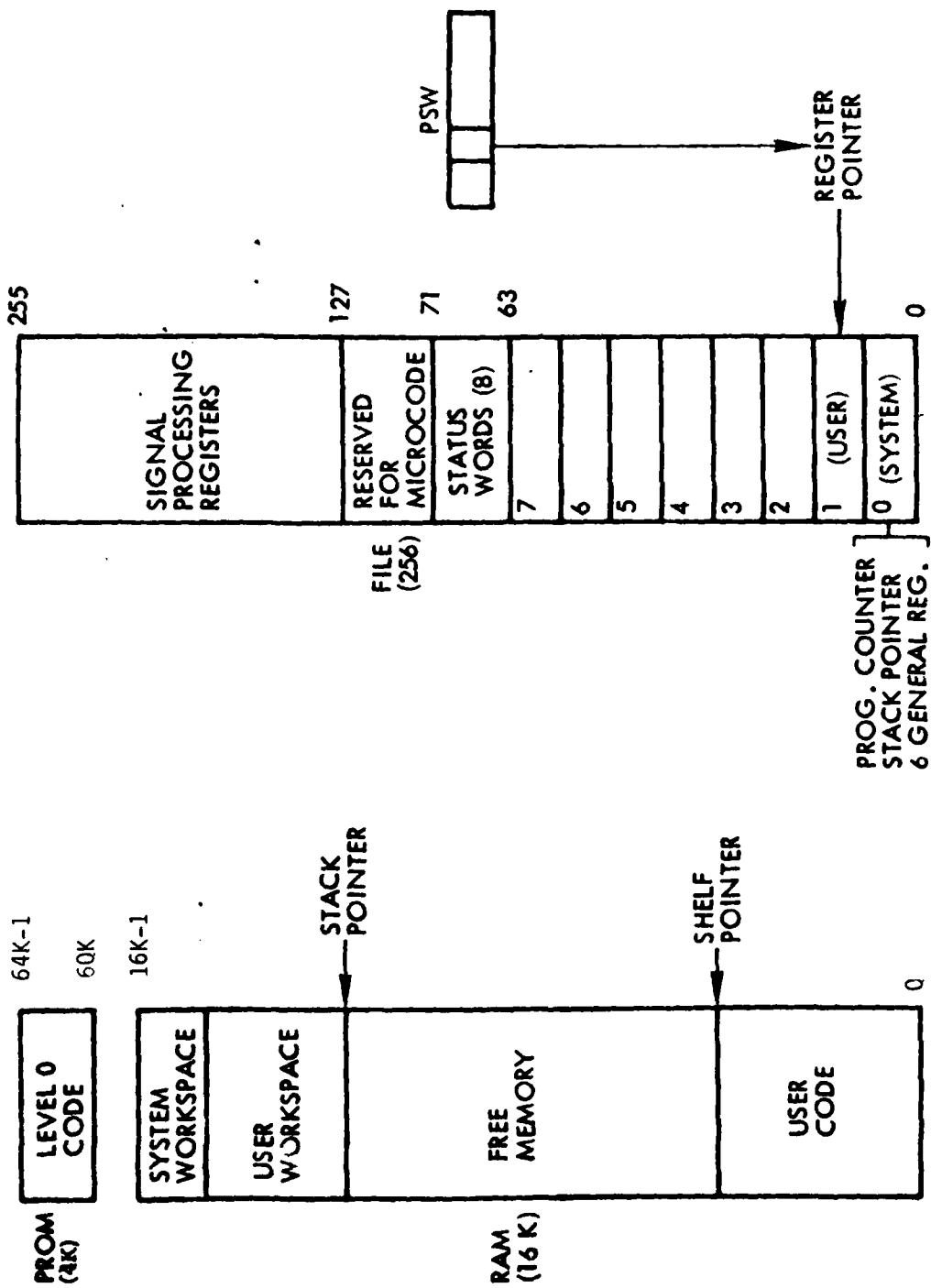


Figure 5-2. Common Element Memory Utilization

Table 5-1. CE Task Prologue

<u>Word Number</u>	<u>Contents</u>
0	Task Number [*]
1	Initialization Entry Address [*]
2	Starting Address - Initial Load [*]
3	Unsolicited Input Entry Address [*]
4	Clock Interrupt Entry Address [*]
5	Reconfiguration Entry Address [*]
6	Starting S Value ^{††}
7	Socket Address ^{††}
'10	Global Data Size [*]
'11 - '27	Base Register Values ^{*††}
'30	Unsolicited Input Options ^{*†}
	Bit 15 - Accept Data
	Bit 14
	• Set: Data + Headers → Data Buffer
	• Clear: { Data → Data Buffer Headers → Header Buffer
'31	Data Buffer Address ^{*†}
'32	Header List Buffer Address ^{*†} } Relative to DORG 0
'33	Unused
'34	Clock Option ^{*†} { Bit 15 Set - Clock Interrupt Desired Bits 7-0 - Clock Interrupt Frequency
'35 - '36	Clock Period ^{*†} (LSB approximately 2 ms)

(Continued on Next Page)

* Set by User at Assembly Time

† Modified by DOS-0 at User Request

†† Modified by DOS-0 for Operating System Usage

Table 5-1. CE Task Prologue (continued)

<u>Word Number</u>	<u>Contents</u>
'37	Trace Indicator ^{*†} { Bit 15 - Set - Start Trace Clear - Stop Trace Bits 7-0 - Trace Frequency
'40	Modify Virtual Address Indicator Bit 15 = 1 Modify Permitted = 0 No Modify Permitted
'41	PSW Values for Initialization Entry
'42	PSW Values for Starting Address
'43	PSW Values for Unsolicited Input Entry
'44	PSW Values for Clock Interrupt Entry
'45	PSW Values for Reconfiguration Entry
'46 - '47	Unused
'50	Direct I/O Entry Address Message Code 5 [*]
'51	Direct I/O Entry Address Message Code 6 [*]
'52	Direct I/O Entry Address Message Code 10 [*]
'53	Direct I/O Entry Address Message Code 11 [*]
'54	Direct I/O PSW Value Message Code 5
'55	Direct I/O PSW Value Message Code 6
'56	Direct I/O PSW Value Message Code 10
'57	Direct I/O PSW Value Message Code 11
'60 - '77	Trap Locations [*] /Return Addresses ^{††}
'100	Data Recording - Base Extraction Point Number ^{*††}
'101	Data Recording - On/Off Flag Word ^{*††}
'102 - '377	Reserved for Expansion

Must be 0 If Not in Use
Must be 2 If Entry in Use

* Set by User at Assembly Time

† Modified by DOS-Ø at User Request

†† Modified by DOS-Ø for Operating System Usage

5.1.3 Input/Output Structure

Inter-element communication in the FTSP is packet oriented. Packets are blocks of information which are from 1 to 256 words in length. Each packet consists of a header of 1 to 8 words, a body word count, and a body of 0 to 254 words (See Figure 5-3).

The header word(s) format is shown in Figure 5-4, and consists of a 6-bit destination virtual address, a 6-bit source address, and a 4-bit message code which defines the purpose and contents of the packet. The number of header words is determined by whether the packet must pass between clusters. For intra-cluster messages, only one header word is used. Otherwise, 8 words are supplied, with unused words set to 0. In FTWRP, only intra-cluster communications are supported.

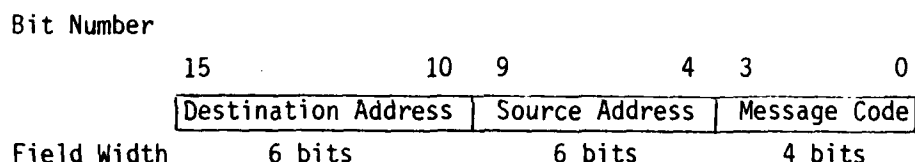


Figure 5-4. Message Header Format

Table 5-2 describes the various message codes, their purposes, valid uses, and message format numbers for each. The format numbers are described in Table 5-3 which is cross-referenced back to Table 5-2. It should be noted that the message codes as listed in these tables are described as they relate to their usage by CEs, IOCs, and CMs. The Intecolor intelligent terminal also accepts these message codes, but with somewhat different formats and/or functions. A complete description of Intecolor supported messages may be found in Section 5.2.2.1.2.

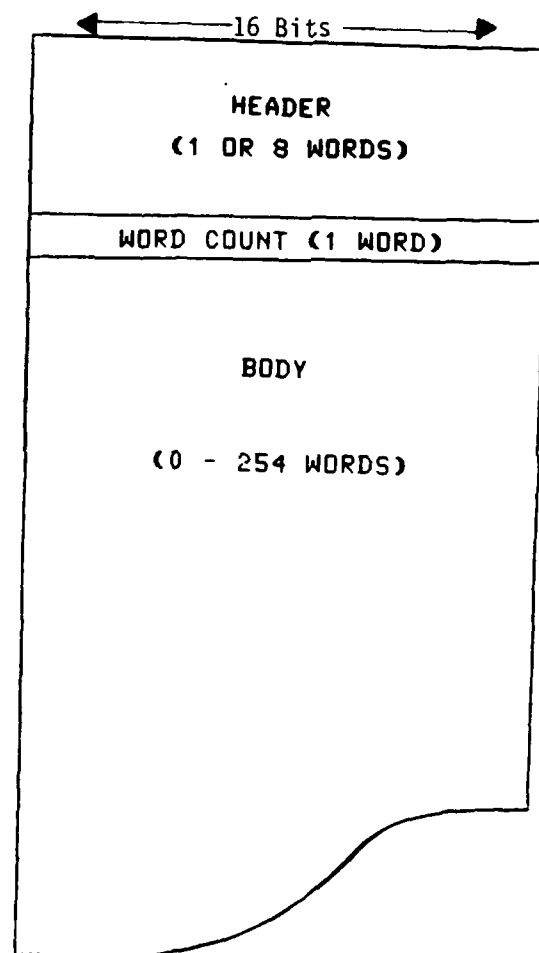


Figure 5-3. FTSP Message Packet Format

Table 5-2. Message Code Definitions

<u>Code</u>	<u>Definition</u>	<u>Valid To</u>	<u>Valid From</u>	<u>Format</u>
0	First Block of Multi Block Message	CE	CE, IOC	1
1	Intermediate Block of Multi Block Message	CE	CE, IOC	1
2	Last Block of Multi Block Message	CE	CE, IOC	1
3	Single Block Message	CE	CE, IOC, CM	1
4	Returned Block	CE	IOC*	1
5	Input Request (Fetch)	IOC, CM	CE, IOC	2 (IOC) or 3 (CM)
6	Output Request (Store)	IOC, CM	CE, IOC	1 (IOC) or 4 (CM)
7	Status Return	CE**	CE, IOC, CM	1
8	Status Request	CE, IOC, CM	CE***	5
9	Fault Message	CE**	CE	1
10	Load Control Word	IOC, CM	CE***	6
11	Bus Extension	IOC*	CE, IOC, CM	n/a
12	Load Virtual Address	IOC, CM, CE	CE***	8
13	Reset and Power On	IOC, CM, CE	CE***	5
14			Unassigned	
15	Power Off (Permanently)	IOC, CM, CE	CE***	5

* Bus Extender IOC only.

** Executive CE only.

***Only from Executive CE.

AD-A106 253

RAYTHEON CO WAYLAND MA EQUIPMENT DIV

F/G 17/4

R&D EQUIPMENT INFORMATION REPORT. FAULT TOLERANT WEATHER RADAR --ETC(11)

MAR 81 M J YOUNG, A J JAGODNIK

F1962A-7A-C-0113

UNCLASSIFIED

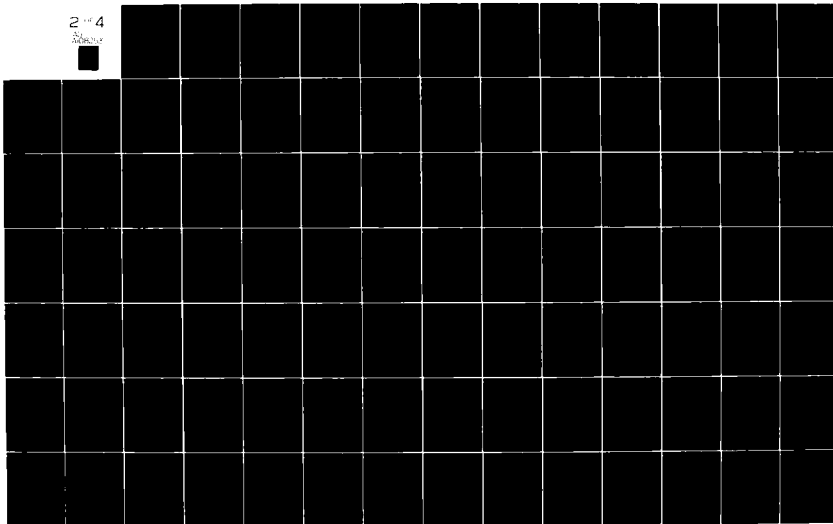
ER81-4053

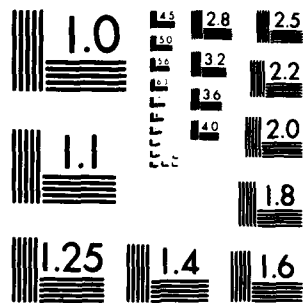
AFGL-TR-81-0086

NL

2-4

2-4





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

Table 5-3. Message Formats

Format Numbers	Message Codes	Message Format
1	0-4, 6(IOC), 7, 9	Header 1, ..., Header N, Message Word Count = M, Data, ..., Data _M
2	5(IOC)	Header 1, ..., Header N, Message Word Count, Requested Word Count
3	5(CM)	Header 1, ..., Header N, Message Word Count, Page Number
4	6(CM)	Header 1, ..., Header N, Message Word Count, Page Number, Data, ..., Data _M
5	8, 13, 15	Header 1, ..., Header N
6	10	Header 1, ..., Header N, Message Word Count = M, Control Word 1, ..., Control Word M
7	Unused	
8	12	Header 1, ..., Header N, Message Word Count = 1, Virtual Address

N (Number of header words) = 1 or 8
M (Message Word Count) ≤ 247

Table 5-4. System Requests -DOS-0

<u>Request #</u>	<u>Data Packet Format</u>
0 Write	<div> Word 0 <div> Status Word <div> <div> Bit 15 Request Queued </div> <div> Bit 12 I/O Completed </div> <div> Bits 11, 10 11 - I/O Error 10 - Data Management Error </div> <div> Bits 7 - 0 11 - Status Byte 10 - Code </div> </div> </div> </div>
1 Read	
	Word 1 Device Number (or Header List Address)
	Word 2 Word Count
	Word 3 Buffer Address (absolute address)
	Word 4 Options
	<div> <div> Bit 15 Executive Message (output only) - Uses Word 7 </div> <div> Bit 14 No Retry on Recoverable Errors </div> <div> Bit 13 Device = Header List Address (output only) </div> <div> Bit 12 Scatter Read (input) (uses Bits 7 - 0) </div> <div> Bit 11 System Packet (Executive only) </div> <div> Bit 10 Associated Input Request (Exec only) </div> <div> Bit 9 Common Memory Request </div> <div> Bit 8 Multiple Input Request </div> <div> Bit 7 0 → Complex Scatter Read 1 → Real Scatter Read </div> <div> Bits 6 - 0 Scatter Read Interval (1 - 127) </div> </div>
	Word 5 Common Memory Page # or Multiple Input Request Parameter
	<div> <div> Bits 15 - 8 Number of Requests </div> <div> Bits 7 - 0 Address Delta </div> </div>
	Word 6 Associated Input Entry Address (DOS-0 use only)
	Word 7 Executive Header Skeleton (output only) (Must contain message code - may contain destination V A)

Table 5-4. System Requests -DOS-0 (continued)

<u>Request #</u>	<u>Data Packet Format</u>	
2 Trace Update	Word 0	<p>Bit 15 Clear - Stop tracing Set = Start tracing</p> <p>Bits 7 - 0 Trace frequency interval</p>
3 Unsolicited Input Update	Word 0	New Options Word (see Prologue for Definition)
	Word 1	New Data Buffer Address
	Word 2	New Header Buffer Address
		} Relative to DORG 0
4 Clock Interrupt Update	Word 0	New Options Word
	Words 1 - 2	New Clock Period
5 DOS-1 Task Directives	Word 0	Directive Types
		<p>1 = Schedule task</p> <p>2 = Suspend task</p> <p>3 = Resume task</p> <p>4 = Abort task</p> <p>5 = Swap tasks</p>
		For Types 1, 2, 3, 4
	Word 1	Task number
	Word 2	Starting Address for Task Execution (If 0, use address in prologue)
		For Type 5 (Swap Tasks)
	Word 1	<p>Bits 15 - 8 Virtual Address 1 of Swap</p> <p>Bits 7 - 0 Virtual Address 2 of Swap</p>
	Word 2	Starting Address for Virtual Address 1
	Word 3	Starting Address for Virtual Address 2
6 Register User Fault	Word 0	User Fault Number (0 to 15)

Table 5-4. System Requests -DOS-0 (continued)

<u>Request #</u>	<u>Data Packet Format</u>
7 Data Recording Request	<p>Word 0 Extraction Point Number</p> <p>Word 1 Number of Subrecords</p> <p> Each Subrecord has Format:</p> <p>Word 2 Number of Words (Subrecord 1)</p> <p>Word 3 Absolute Address of Data to be Recorded</p> <p>Word 4 Number of Words (Subrecord 2)</p> <p>Word 5 Absolute Address of Data to be Recorded</p> <p> : :</p>
8 Update Recording Control Words	<p>Word 0 Logical Device Number (see Section 8)</p> <p>Word 1 Recording On Flags</p> <p>Word 2 Recording Off Flags</p>
9 Modify Virtual Address	Data Packet Address = New Virtual Address
10 Dequeue Output Request	Data packet format same as for Request 0
11 Dequeue Input Request	Data packet format same as for Request 1.

Table 5-4. System Requests - DOS-0 (continued)

<u>Request #</u>	<u>Data Packet Format</u>
12 Standing Output Request	Packet format same as Request 0
13 Standing Input Request	Packet format same as for Request 1
14 Direct Output	Word 0 Status Word Word 1 Header List Address Word 2 Word Count Word 3 Data Buffer Address

5.1.4 System Call Processing

This portion of DOS-0 responds to requests issued by user code for certain system services best provided by the operating system. The processing is flexible enough to provide for up to 32 unique calls, for which the user supplies an address of a packet containing detailed information on the desired service. There are currently fourteen distinct system requests which are honored by DOS-0 (See Table 5-4).

5.1.4.1 Write (0) and Read (1) Requests

When a user makes a request for I/O, the status word of the user packet (Word 0) must be set to 0. After the request is made, the status word must be checked in order to verify that the request was queued. This is done by testing Bit 15; if set, the request has been queued and the I/O will be attempted. If not, DOS-0 was unable to find space in its data base for the I/O request.

Once a request has been queued, the status may be checked by interrogating the I/O complete bit in the status word (Bit 12). Until all of the requested I/O has been completed or an error has occurred, this bit remains 0. Thus the user should periodically check this bit. Once it is set, the determination of correct or incorrect termination of the I/O request is determined through the two error bits, 11 and 10.

Bit 11 set indicates an error has occurred in the attempt to transmit or receive data. If Bit 11 is set, the bottom byte of the status word (Bits 7 - 0) contains the I/O status word from which can be obtained the actual error information. (see Section 4.1.3.1.8).

Bit 10 set indicates an error has occurred in DOS-0's attempt to manage the data represented by this request. In this case, the bottom byte contains a code indicating which type of error occurred; e.g., word count too large, etc.

If neither error bit is set, the I/O request has been completed as requested and the user may now reuse the packet for other requests.

If a user request for input generates an output request, and an error occurs in that output processing, both the output request and input request packets are set to indicate an error, which would be the same error in both packets.

An I/O request, once queued, may be dequeued from DOS-0 by issuing one of the dequeue system requests. DOS-0 responds to the user indicating that either the request was properly dequeued (bit 14 of status word), or that the request was not found in the DOS-0 queues (bit 13 of status word).

Standing order I/O (options bit 7) permits one-time registering of an I/O request with subsequent enabling of the request with modifications to word count, buffer address and page number. I/O overhead is reduced through this approach. Enabling of a suspended standing order request takes the form of either clearing the status word of the original system request packet, or by issuing either system request 12 (output) or system request 13 (input).

5.1.4.1.1 Option Bits (packed word 4)

Executive Message (Bit 15) (Output Only)

Used for output only. This option permits the modification of the header list of the device selected by OR'ing in the executive header word (Word 7 of packet) into the destination header word. This skeleton must contain a valid message code and may or may not contain a destination address, depending on the device.

No Retry on Error (Bit 14) (Output Only)

If this bit is set and a recoverable error is encountered (e.g., bus busy or receiver busy), the system will not retry the I/O.

Header List Supplied (Bit 13) (Output Only)

Use of this option directs DOS-0 to bypass the device-number-to-header-list translation phase of I/O processing by using the header list address supplied in the device word field of the user packet. Use of this option requires that both the destination field and the message code field appear in the header list.

Scatter Read (Bit 12) (Input Only)

If this bit is set, the operating system request performs a scatter read of data, with each word read being separated from the next by a count equal to Bits 5 - 0 of the options word. For example, to scatter read into every 9th word, set Bit 12 and store a 9 into the lower bits. Bit 6 is then used to indicate real (set) or complex (clear) read, i.e., complex reads two words, skip, read two words, etc.

System Packet (Bit 11)

When this bit is set, the DOS-0 system has invoked a packet from the packet stack. In the case of I/O message completion, the packet must be returned to the packet stack.

Associated I/O Request (Bit 10) (Output Only)

When set, this bit indicates that the current I/O request is related to an I/O request to a common memory or a non-bus-extender IOC. In this case, the PAEA field of the I/O packet must contain the I/O queue entry address for the associated request.

Common Memory Request (Bit 9)

If set, the bit indicates that the CM page field of the packet contains a page number to be fetched or stored.

Multiple Input Requests (Bit 8)

This option automatically regenerates the specified number of input requests with each input buffer address incremented by the specified amount. It uses Word 5 for number of requests and address delta.

Standing Order I/O Request (Bit 7)

This option queues up and suspends an I/O request, permitting efficient enabling of the request at a later time.

Direct Output Request (Bit 6)

This option permits direct execution of an output message, where the user supplies control information (header list) and performs error recovery.

5.1.4.1.2 Logical Devices

Under normal conditions, I/O is logical device oriented--that is, the user supplies (via word 1 of the I/O packet) the logical device number of the unit to be accessed, rather than the physical or virtual address data. This permits the user to perform I/O without considering device-dependent problems (e.g., an input request to a common memory requires a message to be output first--this is performed by DOS-0 transparent to the user).

The logical devices as they are defined in FTWRP are listed in Table 5-5. DOS-0 maintains a configuration table (which can be updated by DOS-1 at any time) which maps the logical device number with the various physical requirements of each unit in the system.

5.1.4.1.3 Executive Header Skeleton

Normally, all messages output use message code 3 (single block message). If a non-standard message code is to be used, bit 15 in the I/O options word is set and the desired message code is placed in word 7. A complete description of the various message codes and their functions is given in Section 5.1.3.

5.1.4.2 Trace Update Request (2)

This request changes the trace options words in the user task prologue as requested in the first word of the request packet.

5.1.4.3 Unsolicited Input Update (3)

If a message is received that has no corresponding input request entry (i.e., from a previous system request 1), it is treated as an unsolicited input. This system request instructs DOS-0 what to do with the message before calling the user's unsolicited input service routine. If the options word specifies that unsolicited inputs are no longer allowed, DOS-0 treats the message as an error and reports the condition to DOS-1.

5.1.4.4 Clock Interrupt Update (4)

This request is used to turn the user clock interrupt on or off, and sets the period, which is determined by the formula

$$\Delta t_u = 3\left(-\frac{n}{2} + 1\right) \Delta t_r$$

where n is the number entered in words 1 and 2 of the packet. The refresh period, Δt_r , may vary from CE to CE, but will be between 1.6 and 2.1 mS (1.8 mS nominal). Turning off the clock has no effect on the DOS-0 system clock.

Table 5-5. FTWRP Logical Device Numbers

<u>Logical Device</u>	<u>Virtual Address</u>	<u>Definition</u>
0	77	Program Load Device
1	77	Bootstrap Load Device
2	77	System Device (IDOS-1)
3	77	Operator Device
4	77	Trace Device
5	-	Undefined
6	-	Undefined
7	-	Undefined
8	67	Waiting Signal Processor Task
9	70	Diagnostics Task
10	51	Radar Data Source
11	-	Undefined
12	50	SP Data Output
13	60	Radar Signal Processor 1
14	61	" " " 2
15	62	" " " 3
16	63	" " " 4
17	64	" " " 5
18	65	" " " 6
19	66	" " " 7

5.1.4.5 DOS-1 Task Directives (5)

This request is used as a user interface to the DOS-1 task directives to DOS-0. The feature is supported but not utilized in FTWRP.

5.1.4.6 Register User Fault (6)

This request provides a mechanism by which a user may report faults to DOS-1 via the DOS-0 status words. The fault is reported to DOS-1 at the next status request. After the fault is reported, the condition is automatically cleared. Faults 0-7 are treated as nonfatal, whereas faults 8-15 are reported as fatal (i.e., DOS-1 aborts the task upon receipt of the fault message).

5.1.4.7 Data Recording (7,8)

These services deal with the means by which DOS-0 automatically extracts data from arrays to be output to a recording device. This feature is not supported in FTWRP.

5.1.4.8 Modify Virtual Address (9)

System Request 9 is a special feature installed specifically for the FTWRP system. By calling this service, a task can instruct DOS-0 to modify its own virtual address without direct intervention of DOS-1. This feature is used in the spare rotation algorithm of FTWRP (see Section 5.6).

Authorization for DOS-0 to change its address is given either by a message from DOS-1 (which must be received before the system request can be executed), or by a special authorization word in the user task prologue (see Table 5-1). If no previous authorization was given, DOS-0 reports a fault to DOS-1 and drops to its idle state (i.e., aborts the task).

5.1.4.9 Delete I/O packets (10,11)

These requests allow a mechanism by which pending I/O can be aborted in case the need has disappeared. For example, a pending input request can be aborted if (because of a fault) the input doesn't come in within a specified time limit, or it is no longer needed.

5.1.4.10 Standing I/O Requests (12,13)

One of the original problems associated with packet I/O is the high operating system overhead associated with queueing packets and decoding logical devices, etc. By permitting standing I/O, the packet associated with a message is no longer deleted when I/O is complete. Therefore, next time the same message request is required, the old packet may be re-used, thus eliminating a large amount of overhead processing.

5.1.4.11 Direct Output (14)

This feature provides the ultimate in overhead savings, by requiring the user to handle all header list generation and status processing. If when a direct output request is issued, an attempt fails, it is aborted and an appropriate status is returned. No queue is maintained for these requests. If the output attempt is successful, the transmitter status is returned with no interpretation performed by DOS-0. It is up to the user to check for conditions like bus busy, receiver busy, etc.

5.1.5 Trace Handling

The Common Element firmware supports a trace feature in which an interrupt is generated each time a new instruction is fetched. To enable this feature the trace bit is set in the PSW.

When the Trace interrupt occurs the DOS-0 trace processing service routine performs three checks. First of all, if the task status indicates that the CE is idle, trace processing is aborted. If a task is indeed executing, DOS-0 checks the task prologue to see if the task trace options are enabled. If not, trace processing is aborted. The trace options are modified dynamically via system request 2 (Section 5.1.4.2). If the task trace option is enabled, the trace instruction counter is decremented and checked for zero. If nonzero, trace is aborted (this allows the user to trace every nth instruction by setting the trace counter to n). If all of these checks pass, DOS-0 takes the interrupted task's register contents (which were placed on the stack by firmware) and outputs the data to the system trace device (in FTWRP, this is the Intecolor). DOS-0 then enters a delay loop of approximately one half second to ensure that the trace data doesn't overload the trace device.

5.1.6 Clock Handling

The CE hardware and firmware support an interval timer interrupt structure which permits DOS-0 to be invoked at regular intervals. The interval is a multiple of the dynamic RAM refresh timer (which is implemented by a one-shot, and may be from 1.6 to 2.1 ms). DOS-0 uses this feature to regularly perform self-checking and health management. In addition, DOS-0 supports a "user clock" interrupt, in which the user supplies an interrupt service routine which is invoked at intervals specified in the task prologue (see Section 5.1.2.2). The interval is specified in multiples of 2 ms.

5.1.7 Exception Handling

In the event of a processor exception, the user task is immediately interrupted, and the DOS-0 exception handler is invoked. There are currently three exceptions which are supported: illegal instruction, illegal address, and stack overflow.

The illegal instruction exception is self explanatory; the CE attempted to execute a nonexistent opcode, or a nonprivileged user attempted to execute a privileged instruction (such as WRITE or RACOR). A fault message is sent to DOS-1 and the task is aborted.

The illegal address exception is caused when a nonprivileged user attempts to access a protected memory location in a write operation. The protected memory areas include:

The user task prologue	(0-377 ₈)
The DOS-0 data base	(34000-37777 ₈)
The nonexistent RAM	(40000-167777 ₈)
DOS-0 PROM area	(170000-177777 ₈)

A read operation of one of these areas does not cause an exception. If a write exception occurs, the write operation is blocked by hardware (to prevent corruption of data), DOS-0 sends a fault message to DOS-1, and the task is aborted.

The stack overflow exception is invoked whenever the register set level exceeds 7 (or goes below 0) or if the link stack overflows (or underflows). The following Section provides a description of the link stack and its use.

5.1.8 Common Element Link Stack

On most computers, when a subroutine call is invoked the return link information is saved on the user's stack. Unfortunately, in a fault-tolerant system this is not always expedient, since a faulty program could destroy its stack, and return to a nonexistent (or at least unknown) location to continue execution. Therefore, in the Common Element, a special "Link Stack" is maintained by the microcode for subroutine Linkage.

Each time a subroutine call is executed (via the JSUB instruction), the return linkage information (consisting of the return address and the current PSW) is saved on the link stack. The RETURN instruction simply pops this information off the link stack, just as a normal computer would pop the return address off the user stack. However, integrity is maintained by virtue of the fact that the user has no access rights into the link stack (and therefore cannot destroy it).

Each time a RESUME (from interrupt service routine) is executed, the link stack is checked and all return links from subroutines with the same PSW are deleted. This eliminates the need to execute multiple returns from subroutines to get back to the "main" routine of the current "level" of processing before the RESUME can be executed. Although this can be a handy tool for error conditions and things, it is not recommended to resume in this manner, since it represents a grossly unstructured method of programming, and may be difficult to debug.

The link stack occupies the uppermost 64 locations of the 16K RAM address space (37700-37777 octal). It therefore is contained in one of the protected address spaces of memory, and cannot be modified by a nonprivileged user.

5.2 Intecolor Resident DOS-0 (IDOS-0)

IDOS-0 is an Intecolor resident version of the DOS-0 operating system which resides in the Common Element. Its main purpose is to support the basic operations which enable the Intecolor to function as a valid element on the system bus via the Terminal Interface Element (TIE). As such, IDOS-0 is the interface between the user and the TIE.

All user programs (such as IDOS-1) are executed as a "subtask" of IDOS-0 in much the same manner as a task runs under DOS-0 in the CE. The user program should be stored on a disk in drive 0 under the name "IUSER.PRG". The task prologue should begin at location $A100_{16}$, and the program must reside between $A200-DFFF_{16}$ (Intecolor memory usage is shown in Table 5.6).

Programs are loaded and executed under IDOS-0 in the following manner:

- 1) Turn on Intecolor power using switch in back of unit.
- 2) Insert into drive 0 (left drive) a diskette containing "IUSER.PRG" (the user program).
- 3) Insert into drive 1 (right drive) the diskette containing any files used by IUSER, plus files of the form "PAGE.XXX" which are used by IDOS-0 as Common Memory pages (see Section 3.1.3).
- 4) Type (Escape) P to initialize the Intecolor CPU Operating System.
- 5) Type (Escape) D to enter the Intecolor File Control System.
- 6) Type RUN IDOS0 (Return).

IDOS-0 will initialize its own tables including the page map (see Section 5.2.2.1.3), and then load and call "IUSER.PRG". Execution of IUSER will begin at the address specified in the task prologue.

The following sections describe the operation of IDOS-0 in detail. Appendix B lists the major subroutines of IDOS-0 and their functions, and the tables and variables maintained by IDOS-0.

Table 5-6. Intecolor Memory Usage

<u>Address</u>	<u>Usage</u>
0000-7FFF	Intecolor ROM/PROM
8000-9DFF	Intecolor Display RAM
9E00-9FFF	Scratch area for BASIC, CPU and CRT OS
A000-A0FF	Utility scratch pad and user stack
A100-A1FF	User task prologue ¹
A200-DFFF	User task ¹
E000-F4FF	IDOS-0
F500-F5FF	Configuration Table
F600-F7FF	Page sector address map
F800-F9FF	IDOS-0 output buffer
FA00-FBFF	Page 0 buffer
FC00-FDFF	Page 1 buffer
FE00-FFFF	Page 2 buffer

¹User program should be stored on drive 0 under the filename "IUSER.PRG". Any user required files must be loaded by user program. System is started from file control system by typing 'RUN IDOS0

5.2.1 IDOS-0 Operation

All user programs in the Intecolor which use the TIC must run as tasks under IDOS-0. At initialization, IDOS-0 loads the file "IUSER.PRG" from disk drive 0. For proper operation, the user program should be assembled to load into locations A100-DFFF₁₆. The first 256 locations are reserved for a task prologue.

5.2.1.1 Task Prologue Description

The IUSER task prologue, as stated above, occupies locations A100-A1FF₁₆. The format of the prologue is shown in Table 5-7 and described in the following sections.

5.2.1.1.1 Card Type (A100)

The card type is specified by the user at assembly time, and specifies to IDOS-0 whether the Intecolor is to appear on the bus as a CE, a CM (Common Memory), or an IOC. Only the two least-significant bits are used. This code does not affect the operation of IDOS-0, it merely is used in the status words maintained by IDOS-0 and reported to DOS-1.

5.2.1.1.2 Task Virtual Address (A101)

The task Virtual Address (VA) is a 6-bit number from 00-77₈ which specifies to IDOS-0, the VA which is to be assigned to the Intecolor at startup. If at any time after initialization the VA of the Intecolor is changed (e.g., via a Load Virtual Address command from DOS-1), the Task VA byte in the prologue will be modified accordingly.

5.2.1.1.3 Task Status (A102-A103)

The Status word is meant to be a means of communicating to IDOS-0 any information necessary to proper operation of the user task. At present, if the most-significant bit of the status word is set (bit 15 or bit 7 of loc. A102), IDOS-0 will execute a reset or restart operation. Otherwise, the word is ignored by IDOS-0.

Table 5-7. Intecolor User Task Prologue

Address	No. Bytes	Contents
A100	1	Card Type (1 = CE, 2 = IOC, 3 = CM) ¹
A101	1	Task virtual address ²
A102	2	Task status ³
A104	2	Starting PC address ^{1,4}
A106	2	Clock interrupt service routine address ^{1,4,7}
A108	2	Initial stack value ^{3,4}
A10A	2	Clock options ⁶ Bit 15 = 1 → enabled = 0 → disabled
A10C	2	Clock period ^{2,4}
A10E	2	Page 0 update interrupt address ^{1,4,5,7}
A110	2	Page 1 update interrupt address ^{1,4,5,7}
A112	2	Page 2 update interrupt address ^{1,4,5,7}
A114	2	"Unsolicited Input" interrupt address ^{1,4,5,7}
A116	2	User restart address ^{1,4,5}
A118	1	Display protect boundary -X ¹
A119	1	Display protect boundary -Y ¹

¹Set by user at assembly time.

²Set by user, may be modified by IDOS-0.

³Modified by IDOS-0.

⁴8080 address format (LO byte, high byte).

⁵If zero, interrupt disabled.

⁶When 8080 interrupts are enabled, IDOS-0 interrupts user task approximately every 9.6 ms. User clock is decremented each time. User is interrupted when user clock = 0.

⁷For all interrupts, user must preserve stack address and resume with a subroutine return ('RET') instruction.

5.2.1.1.4 Starting Address (A104-A105)

The starting address word specifies to IDOS-0 the point at which execution of the user program is to begin at initialization. The value is set at assembly time, and is stored in 8080 address format. Therefore location A104 contains the least-significant byte of address, and A105 the most-significant byte (e.g., address A23F is stored as 3FA2). This is done automatically by the assembler using the DW command.

5.2.1.1.5 Clock Interrupt Address (A106-A107)

This word specifies the starting address of the user routine which is to be called when the user clock reaches zero. The user clock will be discussed further in Section 5.2.1.2.2.

5.2.1.1.6 Initial Stack Value (A108-A109)

This word is set by IDOS-0 at load time, and specifies to the user the first location reserved for the user stack. The stack pointer (SP) is also set to this value before execution begins. The user should always use this value when clearing the stack, to be sure that no unauthorized areas are destroyed. This can be done with the instructions:

```
LHLD    0A108H    ;LOAD STACK VALUE
SPHL                      ;STORE IN SP
```

5.2.1.1.7 Clock Options (A10A-A10B)

The Options word is set by the User, and tells IDOS-0 whether clock interrupts are enabled or disabled. This word may be modified by the user directly during execution, or via a system service (see Section 5.2.3).

5.2.1.1.8 Clock Period (A10C-A10D)

If clock interrupts are enabled, this word specifies the length of time between interrupts. As with the options word, the period may be changed directly or via a system service.

5.2.1.1.9 Page 0, 1, and 2 Update Address (A10E-A10F, A110-A111, A112-A113)

Pages 0, 1, and 2 have special significance in some systems, and are utilized quite often. Therefore they are stored in Intecolor memory rather than on disk, and whenever one is modified by a Common Memory write operation, the user may be interrupted. These three words specify the starting address of the three routines which may be called. If the starting address is zero, the interrupt for that page is disabled.

5.2.1.1.10 Unsolicited Input Interrupt Address (A114-A115)

Certain incoming messages must be processed by the user. IDOS-0 makes this possible by providing a user interrupt for this purpose. This word specifies to IDOS-0 the starting address of the user routine that is to be called. If the address is zero, this interrupt is disabled.

5.2.1.1.11 Restart Address (A116-A117)

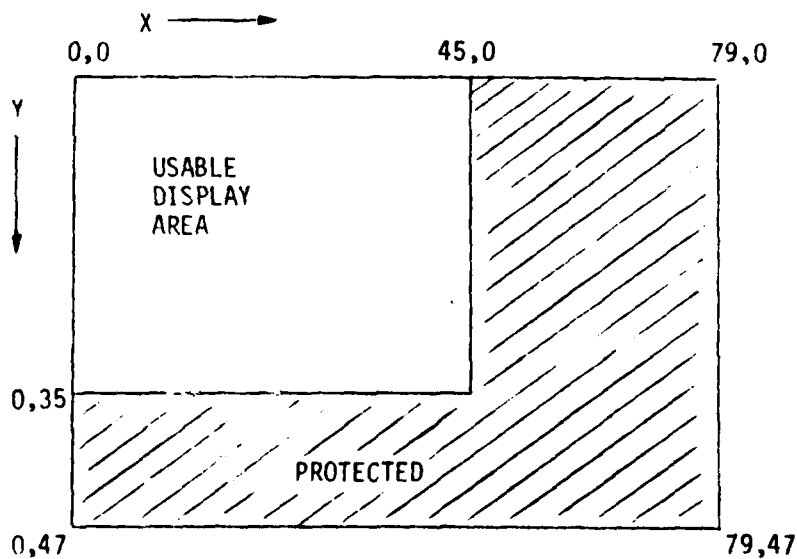
Upon the execution of a reset command (either externally or from the user via the status word--see Section 5.2.1.1.3), the user may want to resume processing at a different location than at initialization. If non-zero, IDOS-0 will not reload IUSER, but will simply vector to the specified address. If zero, IUSER will be reloaded and execution will begin at the address specified by the Starting Address word (A104-A105).

5.2.1.1.12 Display Protect Boundaries (A118-A119)

IDOS-0 has a feature which allows an external source to print text on the Intecolor display without user intervention. Locations A118-A119 specify the acceptable area of the display for this purpose. IDOS-0 will not allow external text to overwrite protected areas of the screen. If the boundaries are set to zero, no text will be permitted (see Figure 5-5).

5.2.1.2 IDOS-0/TIE Interaction

The software which is supplied with the Intecolor uses seven of the eight available interrupts. The eighth interrupt is used by interval timer #1, and is set whenever the timer counts down to zero. IDOS-0 uses this timer to interrupt the user program periodically. The interval is set to approximately 10 ms.



- A118 - X boundary = 45_{10}
- A119 - Y boundary = 35_{10}

Figure 5-5. Text Boundary Protection

5.2.1.2.1 TIE Polling

Each time IDOS-0 is invoked by the timer interrupt, the TIE status is polled. If the TIE is found to be idle, control returns to the user, and little time is lost. If the receiver buffer is found to contain a message, IDOS-0 processes the message and takes appropriate action before returning control to the user.

5.2.1.2.2 User Clock

In addition, each time IDOS-0 is invoked, the user clock is decremented. If the clock value reaches zero, and clock interrupts are enabled, the user clock interrupt routine is invoked before control reverts to the user. This feature permits the user to perform operations at regular intervals.

5.2.2 Input/Output Operation

Because of the architectural differences between the Intecolor and CEs, I/O is handled differently by IDOS-0. The following sections describe the I/O processes and data structures.

5.2.2.1 Input Message Handling

Upon completion of a TIE input operation, the TIE sets the RXINT line (see Section 4.3). At the next interval timer interrupt, IDOS-0 polls the RXINT line, and, finding it set, reads the TIE status byte to determine the cause. If it finds that a successfully completed transfer has taken place, it reads the data from the TIE into Intecolor memory and a buffer packet is constructed. The following sections describe the packet format and the different actions taken for each message code.

5.2.2.1.1 Input Buffer Packets

There are a total of three input buffers maintained by IDOS-0 which are used in a round-robin fashion. This is designed to allow processing on one buffer while others are pending. A pointer (IBUFP) keeps track of the next available buffer for input. Another point (UBUFP) points to the next buffer available to the user. The format of the packets are shown in Table 5-8.

The first byte (byte 0) is a flag denoting the status of the buffer (0 = empty, -1 = full). Bytes 1 and 2 are a pointer to the location where the data is stored. Byte 3 is the number of header words contained in the message. These words are stripped off the message and stored separately from the data in the location pointed to by bytes 4 and 5. Byte 6 gives the virtual address from which the message originated, and bytes 7 and 8 point to the header word in which the source address occurred. Byte 9 specifies the data word count (the number of 16-bit data words). Byte 10 specifies the message code, and the eleventh byte contains the Common Memory Page number, if applicable (message codes 5 and 6).

Table 5-8. IDOS-0 Input Buffer Packet Structure

<u>Word #</u>	<u>No. Bytes</u>	<u>Contents</u>
0	1	Buffer full flag (-1 → full)
1	2	Data buffer address ¹
3	1	Header count
4	2	Header list address ¹
6	1	Source virtual address
7	2	Address of header containing source ¹
9	1	Word Count
10	1	Message code
11	1	CM page number

¹8080 Address Format (LO byte, high byte)

5.2.2.1.2 Message Codes and Functions

IDOS-0 handles incoming messages slightly differently than DOS-0 in a CE. The following sections describe the message codes and the action taken by IDOS-0. Table 5-9 lists them all for reference.

5.2.2.1.2.1 Message Codes 0, 1, 2, and 3

These codes are used to send messages from task-to-task in the CE system. IDOS-0 passes the data directly to the user in the following manner: The current buffer is constructed, and the "full" flag is set. The header is placed in an appropriate buffer, and the data is stored in the current data buffer. Then, if the user prologue has a nonzero value in the unsolicited input interrupt address ($A114-A115_{16}$), the user interrupt routine is called to process the packet. The user then requests the packet, (see Section 5.2.3), processes the data, and releases the buffer before returning control to IDOS-0.

If the interrupt address is zero, the interrupt is disabled, and IDOS-0 immediately returns from the polling routine to the interrupted program. The user may poll the input buffers by requesting a packet and checking the buffer "full" flag. Section 5.2.3 describes this procedure further.

5.2.2.1.2.2 Message Code 4 (Returned Message)

A returned message causes a fault message to be sent to $VA\ 77_8$ (DOS-1), and the packet is released. The user task is not affected.

5.2.2.1.2.3 Message Code 5 (Common Memory Input Request)

Message code 5 is a request for IDOS-0 to fetch a specific page of Common Memory (drive 1 of the floppy disk) and return it to the source address. IDOS-0 performs this function regardless of what card type the Intecolor appears to be on the CE bus. Pages 0, 1, and 2 are maintained in the Intecolor memory itself, at addresses $FA00-FBFF_{16}$, $FC00-FDFF_{16}$, and $FE00-FFFF_{16}$ respectively. All other pages are stored on disk and must be loaded into RAM before output.

Table 5-9. IDOS-0 Message Code Processing

<u>Message Code</u>	<u>Meaning</u>	<u>Function</u>
0	First Block	Passed to user program ¹
1	Middle Block	Passed to user program ¹
2	Last Block	Passed to user program ¹
3	Single Block	Passed to user program ¹
4	Returned Block	Fault message sent to DOS-1
5	Input Request	Performs common memory input request ²
6	Output Request	Performs common memory output request ^{2,3}
7	Status Return	If TSKVA = 77 or 76, passed to user program ¹ ; otherwise, fault message is sent to DOS-1
8	Status Request	If source = 77, status is sent; otherwise, fault message is sent to DOS-1
9	Fault Message	Passed to user program ¹
10	Text Message	Message displayed on screen
11	Bus Extender	Fault message sent to DOS-1
12	Load Virtual Address	If source = 77, VA is changed in TIE and task prologue; otherwise fault message sent to DOS-1
13	Reset	IDOS-0 reset, control transfers to initial entry point
14	Executive Message	If source not 77, fault message sent to DOS-1; otherwise ignored unless configuration table update, in which case only first word is used to derive transmit bus commands
15	Power OFF	IDOS-0 reset

¹Passed via circular input buffers; if unsolicited input enabled, user is interrupted.

²Regardless of card type.

³If pages 0, 1 or 2 are updated, user is interrupted (if enabled); all other page requests are ignored.

To speed up access to the floppy disk, a "page map" which contains the starting sector of each page is constructed and maintained by IDOS-0. This page map is discussed further in Section 5.2.2.1.3.

5.2.2.1.2.4 Message Code 6 (Common Memory Output Request)

This code is used to store a block of data into a page of Common Memory. IDOS-0 replaces the old page with the data in the input buffer. If the page number did not previously exist on the disk or in the page map, an error results, and no replacement takes place. The old version of the page is overwritten, and may no longer be used after this operation. Updates to pages 0, 1, and 2 are performed in memory, and do not involve a disk access.

(At present in FTWRP, only pages 0, 1, and 2 may be modified. Any request to update a page on disk is ignored by IDOS-0.)

If page 0, 1, or 2 is updated, and the corresponding interrupt is enabled in the task prologue, the user is called before the packet is released, and before processing resumes.

5.2.2.1.2.5 Message Code 7 (Status Return)

Code 7 is legal only if the VA of the Intecolor is 77_8 (i.e., the user program is IDOS-1) or 76_8 . In this case the buffer is passed to the user as in Section 5.2.2.1.2.1. Otherwise the packet is released and a fault message is sent to DOS-1.

5.2.2.1.2.6 Message Code 8 (Status Request)

If the source of this message is 77_8 or 76_8 , IDOS-0 will output the four IDOS-0 status words with a message code 7 (Status Return). After output is complete, the status bits are cleared and the input buffer released. If the source is anything other than 77_8 or 76_8 , a fault message is sent to DOS-1.

5.2.2.1.2.7 Message Code 9 (Fault Message)

This message is always passed directly to the user as described in Section 5.2.2.1.2.1 regardless of who the source is or what the Intecolor's VA is. This is so that the system operator can see fault messages on the display and take appropriate action.

5.2.2.1.2.8 Message Code 10 (Text)

This code is a special case in IDOS-0. Any block that comes in with message code 10 is treated as text by IDOS-0 and output to the screen. The user task is unaffected and is not called. The format of code 10 messages is shown in Figure 5-6. Before the text is output, IDOS-0 compares the cursor location on the screen with the boundaries of usable display specified in the task prologue. Any characters which place the cursor in a protected area of the display are suppressed. Certain special characters (such as Escape) are also suppressed to prevent an external source from making the Intecolor go wacky. A complete list of valid and invalid characters is shown in Figure 5-7.

5.2.2.1.2.9 Message Code 11 (Bus Extender)

This code is always illegal to the Intecolor, and IDOS-0 will log a fault message to DOS-1 if it is received.

5.2.2.1.2.10 Message Code 12 (Load Virtual Address)

If the source is 76_g or 77_g, IDOS-0 will change the Intecolor's VA to the value specified in the first data word of the message. The Task VA in the user prologue is also modified. If the source is invalid, the VA is left unchanged and DOS-1 is notified.

5.2.2.1.2.11 Message Codes 13 and 15 (Reset and Power Off)

These codes are not normally seen by the CE because hardware exists to perform the functions immediately. However, the TIE has no such hardware, so they must be done in software. Both codes perform the same function. Upon receipt of a 13 or 15, IDOS-0 aborts the user task, re-initializes its tables, and, if the restart address in the user prologue is nonzero, vectors to that address. If the restart address is zero, the user task is reloaded and started at the initialization address.

<u>Byte Number</u>	<u>Word Number</u>	<u>Contents</u>	<u>Comments</u>
0-1	0	Header Word To: IDOS-0 From: Anyone Message Code: 10_{10}	.
2-3	1	Body Word Count (≤ 252)	
4-5	2	Text Message Character Count	
6-7	3	Start Location	
6		Bits 15-8 Column # (0-79)	
7		Bits 7-0 Line # (0-47)	
8-2n	4-n	Text - 2 characters per word	

Figure 5-6. IDOS-0 Text Message Format

5.2.2.1.2.12 Message Code 14 (Executive Message)

This code is reserved for communication between DOS-1 and IDOS-0, and never involves the user task. The format of the executive message is shown in Figure 5-8. The message types are listed in Table 5-10. At this time, all message types are ignored by IDOS-0 except configuration update (type 2). If a configuration update is received, the second data word is read for transmit bus information. If bit 15 is set, IDOS-0 will alternate busses each time a message is output (first A, then B, etc.). If bit 15 is reset, then bit 14 is used to specify which bus is to be used (0 = A, 1 = B). At present, the rest of the configuration update message is ignored, although in the future the capability of using logical devices may be added.

5.2.2.1.3 Page Map

As stated before, IDOS-0 expects drive 1 to contain files which are treated as Common Memory pages. Each "page" is a self-contained file with the name "PAGE.XXX", where "XXX" is a decimal number from 000 to 255. Thus, PAGE.042 is the name of the file containing page 42. Each page is exactly 240_{10} 16-bit words in length.

To speed up Common Memory operations, IDOS-0 constructs a "page map" in Intecolor memory ($F600-F7FF_{16}$) which contains the starting sector of each page on the disk. The format of the page map, shown in Figure 5-9, consists of two bytes per page. The 512-byte map permits up to 256 pages to be specified. Nonexistent pages are entered as -1 ($FFFF_{16}$).

Upon initialization, IDOS-0 scans the directory of drive 1, looking for filenames of the form "PAGE.XXX". The starting sector address of each page is placed in the appropriate word of the page map. If multiple copies of a page exist, the most recent version is placed in the map. If the disk is modified or replaced, a new page map must be constructed to prevent loss of files. This can be accomplished by a system service supported by IDOS-0 which may be called by the user program (see Section 5.2.3).

Word

	15	10	9	4	3	0
0	DEST			SOURCE		1 1 1 0
1	WORD COUNT					
2	MESSAGE TYPE					
3-n	BODY					

Figure 5-8. Executive Message Format

F600	LOW BYTE OF SECTOR ADDR	HIGH BYTE OF SECTOR ADDR	PAGE 0
F602	-1	-1	PAGE 1
	(PAGE NONEXISTENT)		
F604	LOW BYTE	HIGH BYTE	PAGE 2
F606			.
.			.
.			.
F7FC	-1	-1	PAGE 254
	(PAGE NONEXISTENT)		
F7FE	LOW BYTE	HIGH BYTE	PAGE 255

Figure 5-9. Page Map Format

Table 5-10. Executive Message Types

<u>Message Type</u>	<u>Format of the Message Body</u>	
0 Load Task	Word 1	Load Map Page Number
1 Start Task	Word 1	Task Number
	Word 2	Starting Address or 0
2 Configuration Data Update	Word 1	k Number of Configuration Items plus Bus Control Word
	Bit 15	0 = No Alternating 1 = Alternating
	Bit 14	0 = Bus A 1 = Bus B
	Bits 13-0 Number of items	
	Word 2	1 Number of Words for Item 1 including this word count
	Word 3	d Device Number
	Word 4	VA Virtual Address of Actual Card
	Word 5	m Index to Final Header ($1 \leq m \leq n$)
	Word 6	n Number of Headers
	Word 7-1+1	Header(s)
	Word 1+2 and on for items 2 through k	
3 Task Directive	Word 1	Directive
		2 = Suspend Task
		3 = Resume Task
		4 = Abort Task
4 Memory I/O	(See Reference 5-1)	

Continued on second page...

Table 5-10. Executive Message Types (Continued)

<u>Message Type</u>	<u>Format of the Message Body</u>	
5 Modify Memory	Word 1	Number of Modifications (n)
	Word 2	Address for Modification 1
	Word 3	New value for Address in Word 2
	Word 4	Address for Modification 2
	Word 5	New value for Address in Word 4

	Word 2	Address for Modification n
	Word 2n+1	New value for Address in Word 2n
6 Update Virtual Address Modifi- cation Control Word	Word 1	New Virtual Address Modification Control Word
	Bit 15	<div style="display: inline-block; vertical-align: middle;"> <div style="font-size: 2em; vertical-align: middle; margin-right: 5px;">{</div> <div style="display: inline-block; vertical-align: middle;"> Set: Modification Permitted Clear: Modification Prohibited </div> </div>
7 Update Recording Control Words	Word 1	Recording On Flags
	Word 2	Recording Off Flags

Table 5-11. IDOS-0 Service Calls*

<u>Call Number</u>	<u>Description</u>	<u>Parameters</u>	<u>Returned Values</u>
0	Output Message	B = Header Count C = Data word count DE = Header List address HL = Data buffer address	CARRY = I/O error NO CARRY = I/O successful A = Error code
1	Update user clock interrupt	BC = New options word DE = New clock period	
2	Set virtual address	B = New virtual address	
3	Select Bus	B = Bus 0 = A bus 1 = B bus -1 = Alternate busses	
4	Register Fault	B = Fault number (0-7) If bit 7 set, error set in user fatal status word; task aborted If bit 7 reset, error set in user nonfatal status word; control reverts to user	
5	Fetch Input Buffer Packet	HL = address to which packet is to be stored	A = Buffer full flag (word 0 of packet; -1 = full, 0 = empty) condition codes set
6	Release Input Buffer	No parameters	
7	Initialize Page Map	Reads directory of drive 1 and generates new page map; display RAM (8000-90FF) is used as temp. buffer	

No parameters

Continued on next page...

Table 5-11. IDOS-0 Service Calls* (Continued)

<u>Call Number</u>	<u>Description</u>	<u>Parameters</u>	<u>Returned Values</u>
8	Software Interrupt	Generates an I/O interrupt which polls TIE; clock is not updated. No parameters	

*Executed by a subroutine call to E003H. A register must have service call number; parameters passed in registers as shown above.

Table 5-12. Intecolor Input/Output Error Messages

Input Error Codes

Displayed in the form:

ERROR XX DURING INPUT

Where XX is:

2x	Fault on A Bus
4x	Fault on B Bus
Ax	Parity Error on A
Cx	Parity Error on B
80	Interrupt, Receiver Idle
81	TIE Command Error - Illegal Command
82	TIE Byte Boundary Error
84	TIE Read Reset Error - Receiver Idle
8B	TIE Read Error - Receiver Idle

Output Error Codes

Displayed in the form:

ERROR XX DURING OUTPUT

CALLED FROM YYYY

Where YYYY is Address from which call is made; XX is:

03	Time Out Error
05	Parity Error
07	Receiver Busy
09	No Reply
0B	Arbitration Fault
0D	Bus Busy

Continued on next page...

Table 5-12. Intecolor Input/Output Error Messages (Continued)

Output Error Codes (con't)

0F	Triggered
81	TIE Command Error - Illegal Command
82	TIE Byte Boundary Error
88	TIE Select Bus Error - Transmitter Triggered
89	TIE Set Virtual Address Error - Transmitter Triggered
8A	TIE Write Error - Transmitter Not Idle

5.2.3. System Services

IDOS-0 supports nine system services which the user may call (see Table 5-11). System services are called by placing the service number in the A register, any other parameters in the B, C, D, E, H, and L registers, and executing a subroutine call to E003₁₆. Some services return with a status word in the A register and the flags modified. Each service is discussed at length in the following sections.

5.2.3.1 Output Message (0)

Messages are output by a user program in the Intecolor via service 0. Before the call is made, the B register specifies the number of header words (usually 1 in a single cluster system), and C specifies the number of data words in the message (0 - 253). DE and HL register pairs point to the memory locations containing the headers and data respectively.

After the transfer is complete, IDOS-0 returns to the calling program with a status byte in A and the carry flag modified. If carry is zero (no carry), output was successful. If carry is set, an error occurred, and the A register contains an error code. All errors except No Reply (code 09) are also reported to the operator as an error message in the lower right portion of the Intecolor display. The error codes are defined in Table 5-12.

5.2.3.2 Update User Clock (1)

The user may change the user clock period or options at any time via a service call 1. Before the call, the BC register pair should contain the new options word (0 = interrupts disabled, -1 = enabled), and DE should contain the new clock period. The period is decremented by one each time IDOS-0 is invoked by the interval timer, and thus the actual period of the user clock is a multiple of about 10 msec. However, the clock is very inaccurate, and many factors change its period.

5.2.3.3 Set Virtual Address (2)

Service call 2 takes the value in the B register and stores it in the virtual address register of the TIE. The task VA byte in the user prologue is also updated.

5.2.3.4 Select Bus (3)

Service 3 enables the user to specify which external bus the TIE should use on output. If the B register contains a zero, Bus A is always used. If B is a 1, Bus B is used. If B is a -1, the bus is alternated each time an output is performed. This command overrides the information in a configuration update from DOS-1 (see Section 5.2.2.1.2.12).

5.2.3.5 Register Fault (4)

One of the four status words maintained by IDOS-0 is reserved for user fault reports. The upper byte is reserved for fatal errors, and the lower byte for non-fatal errors.

Register B contains a number from 0 to 7 specifying the bit which is to be set. If bit 7 of the B register is set, the corresponding bit is set in the fatal error byte, and the user program is aborted. Otherwise, the bit is set in the nonfatal error byte, and control is returned to the user. The next time DOS-1 requests status from IDOS-0, the fault will be reported.

5.2.3.6 Fetch Input Buffer Packet (5)

This service transfers the packet for the next available input buffer to the location specified by the HL register pair. In addition, the buffer "full" flag (byte 0 of the packet) is placed in A, and the condition codes set. If the A register is zero, the buffer is empty, and the information in the packet is meaningless. If A is negative (-1), the buffer is active, and the packet information is valid. Room must be reserved for twelve bytes of packet information.

5.2.3.7 Release Buffer Packet (6)

This service takes the current buffer packet and resets the buffer "full" flag, thus releasing it from use. In addition, the user's "current packet" pointer which is maintained by IDOS-0 is incremented to point to the next available packet. This is the method by which a user task informs IDOS-0 that it is through with a packet.

5.2.3.8 Initialize Page Map (7)

As stated in Section 5.2.2.1.3, in order for IDUS-0 to properly perform the Common Memory functions of Input and Output, a page map must be maintained in Intecolor memory. Whenever a new diskette is inserted into drive 1, a new page map must be constructed. IDOS-0 can perform this operation for the user via service call 7. This service uses the Intecolor display refresh RAM as a workspace (8000-9DFF₁₆), and thus destroys anything already on the display. The display should be erased after this call.

5.2.3.9 Software Interrupt (8)

This service is required because of a peculiarity in the Intecolor operating system. Many times during execution of the user program, interrupts (including that of interval timer #1) are disabled by the ISC-supplied software. This is done without warning, and prevents IDOS-0 from polling the TIE to process messages. Thus, at times when the user is waiting for a critical message from the TIE, the input never seems to come. Simply re-enabling interrupts using the 8080 "EI" command does not always solve the problem, and sometimes creates more problems itself. This service permits the user to force IDOS-0 to poll the TIE just as if a timer interrupt had occurred. The only difference between the software and hardware interrupts is that the software interrupt does not affect the user clock.

5.3 Distributed Operating System Level 1 (IDOS-1)

Although the Fault-Tolerant Signal Processor was designed to permit the level 1 operating system to run in a Common Element, in the FTWRP, the 8080-based Intecolor intelligent terminal was the ideal place for it. The name of the program was thus modified to "Intecolor-resident DOS-1", or IDOS-1.

IDOS-1 performs all the functions of a single cluster DOS-1, plus it acts as an operator's console where commands and parameters may be input and system status and command responses are displayed. The following sections describe the system functions of IDOS-1, while the command interpreter is described in detail in Section 3.2.

5.3.1 System Configuration

The current FTWRP system software will support a single cluster of devices including an input IOC, an output IOC and up to 14 Common Elements, plus the Intecolor, which utilizes socket address 0. It should be repeated here that although the software will support a fully populated cluster, electrical power requirements currently limit the maximum number of CEs to 6 (including the test panel, which draws significant power, and two IOCs). More elements will require the addition of a second power source.

5.3.2 FTSP Level 1 Operating System Functions

As stated above, IDOS-1 performs all the functions of a single-cluster version of DOS-1, including system initialization, status polling, task configuration, reconfiguration, spare rotation, and fault reporting. While IDOS-1 is executing, the Intecolor appears on the FTSP busses as a Common Element with virtual address 77 (octal).

5.3.2.1 IDOS-1 Data Structures

A complete list of IDOS-1 data structures and routine usage is contained in Appendix B. The major structures maintained for system configuration are threefold: 1) the system status table (SYSTBL), 2) the system table index (STNDX), and 3) the various system queues. A grasp of the functions of these structures is vital to a sound understanding of IDOS-1 operation.

5.3.2.1.1 System Status Table (SYSTBL)

The SYSTBL is an array of status words, with one entry per physical (socket) address in the system (including the Intecolor). Each entry consists of 8 bytes:

- The card type
 - 0 = nonexistent
 - 1 = CE
 - 2 = IOC
 - 3 = CM
- The current virtual address (0 - 77)
- The card's status (2 bytes)
- The job status (derived from status word)
- The task identification number
- The task virtual address
- The status polling count

The use of each of these words will be discussed in the description of the routines in which they are primarily utilized.

5.3.2.1.2 SYSTBL Index (STNDX)

The STNDX is an array which maps each possible virtual address (0 to 77) to its corresponding socket address, by which the SYSTBL is accessed. For example, if the CE at socket address 10 is executing a task with virtual address 60, the STNDX entry at 60 will contain a 10. This is cross referenced in the SYSTBL via the virtual address word (byte 2), which, in the entry for socket address 10 will contain a 60. Virtual addresses which are unused are flagged by a -1 in the corresponding STNDX entry.

Whenever a card's virtual address (VA) is changed, the new VA entry in the STNDX is given the socket address as it is specified in the old VA entry. The old entry is then changed to -1. Thus IDOS-1 can keep track of a dynamic system without undue headaches.

5.3.2.1.3 IDOS-1 Queues

IDOS-1 maintains a number of queues for system configuration and task loading. The major queues are the Idle list, the Task queue, the Virtual Address queue, and the Spare list. These are all of the form of a last-in first-out stack, that is, the last entry pushed onto the queue is the first one popped off.

The Task Queue contains a list of the identification number of each task which must be loaded next. The priority of task loading is determined by the order in which they are entered into the queue. The VA queue is actually an extension of the Task Queue, and contains the virtual addresses that each task will assume when the task is started. These two queues should therefore track each other exactly, and if they don't an error will be generated.

The Idle List is basically just a list of all the idle CEs in the system at any one time. Whenever the Task Queue contains a task id, the idle list is searched for an idle CE into which the task can be loaded. If one is found, the CE is popped off the idle list, and the task id and virtual address are popped off their respective queues. If a new card is inserted into the system after startup or a task completes, that card's virtual address is entered onto the idle list.

The Spare List is the same as the Idle List, except that CEs which are on the Spare list are not necessarily idle. A spare CE may be executing a task, namely the self-diagnostics task. If a high-priority task must be loaded and there are no CEs on the idle list, the spare list is searched for CEs which can be conscripted for service. If one is found, the diagnostics task is aborted and the card is re-loaded with the new task. Otherwise, the task is left on the task queue until a card is inserted or an existing card completes its current task.

5.3.2.2 IDOS-1 Functional Description

5.3.2.2.1 IDOS-1 Initialization

At startup, IDOS-1 initializes its internal tables and pointers to the initial state. The SYSTBL is initialized such that the only member of the system is the Intecolor itself, which is given the card type of CE, and virtual

address 77 (octal). The STNDX is set to all -ls except the entry at 77, which is set to 0 (socket address 0). Then the Task Queue is loaded from the floppy disk (file name: TASKS.001), along with the VA Queue. In addition, the default system configuration is initialized from disk (file name: CONFIG.001), and the signal processing parameters are loaded into Intecolor RAM from floppy (file name: PARAM.001). After initialization is complete, IDOS-1 enters its idle loop (the command interpreter), waiting for commands from the operator.

5.3.2.2.2 IDOS-1 System Startup

Once the operator has entered all the desired commands (if any), the 'PPP' command is entered to begin system startup.

The first operation performed during startup is to reset all virtual addresses (except 77 octal) to ensure that the system is in a known state. Once this is complete, status request messages are sent to each VA (including 77) to determine the present configuration.

Once a complete list of valid devices is obtained, it is compared with a list of devices necessary to run the desired application. If too few IOCs are available, IDOS-1 enters an infinite loop polling status, waiting for more IOCs to be inserted. This condition is signalled by the error message

NOT ENOUGH IOCS AVAILABLE TO PERFORM TASK
WAITING FOR MORE CARDS

The loop is broken either by the insertion of a sufficient number of IOCs or by depressing the BREAK key. If BREAK is pressed, startup is aborted and the command processor is reentered.

If there are enough IOCs available, the lists are then checked for sufficient CEs. If there are too few CEs, a warning message is displayed to that effect and startup continues.

The next step of system startup is thus the loading of all applications tasks. As each task load operation is complete, the task is started by executive message to the CE. In FTWRP, all signal processing tasks have certain parameter lists which must be loaded before processing can begin. IDOS-1 takes care of this by sending parameter lists to all tasks with virtual addresses between 60

and 67 (octal). In a degraded system, the last entries of the task queue are loaded first (and thus have highest priority). Tasks are loaded until the list of idle (and spare) CEs is exhausted. Inserting more cards will cause the unloaded tasks to be loaded.

Once all the tasks are loaded (or as many as possible in a degraded system), the IOCs are started. IDOS-1 assumes that the input IOC is at socket address 15 (octal) and the output IOC is at 14 (octal). If not, an error is generated. No warning message is displayed, but a fault bit is set in the IDOS-1 status word.

The last operation done at system startup is the enabling of IDOS-1 system clock interrupts, which provides the means to periodically poll system status, etc. Once complete, the command interpreter/idle loop is reentered.

5.3.2.2.3 IDOS-1 Clock Interrupt Processing

At regular intervals (at present, every 87 IDOS-0 clock ticks, or about 1 second), IDOS-0 invokes a clock interrupt routine in IDOS-1 which performs various functions that must be done periodically. Upon entry of this routine (called CLK SVC) a counter is decremented and checked for zero. When zero, this counter signals that system status polling is necessary. Another counter determines when spare rotation must be invoked. Regardless of the states of these counters, the task queue is checked at every clock interrupt. If the queue is not empty, and a CE is found in the idle list (or spare list), the task is loaded at this time.

5.3.2.2.4 IDOS-1 Message Interrupt Processing

Whenever a message is received by IDOS-1, the message processor (MSGP?) is invoked. This routine reads the message, decodes its meaning and takes the appropriate action. The three major message types that are sent to IDOS-1 are message code 3 (single block message), message code 7 (status return), and message code 9 (fault message).

5.3.2.2.4.1 Single Block Messages

Most single block messages are simply displayed on the Intecolor screen with information regarding the source of the message. However, during the course of operation, several messages may come in which serve as synchronization messages for IDOS-1. The meaning of these messages is encoded into the first data word of the body (word 0).

5.3.2.2.4.1.1 No Data (word 0 = 0)

This message, if it comes from a valid signal processing CE (virtual address between 60 and 67), signals that data is not being received from the input IOC. The only time this message should be sent to IDOS-1 is when a CE has been "rotated" out of the processing flow in spare rotation. All other CEs will report this fault as a nonfatal error bit in the status return message.

If the message is invalid because of its source, or reason (e.g., spare rotation is not in progress), the message is displayed as a normal single block message.

5.3.2.2.4.1.2 Look Up Table Synchronization (word 0 = 1)

Whenever a signal processing task is loaded and started in a CE, a parameter list message is sent to it to initialize its internal tables. In addition, a Range Normalization Look Up Table (LUT) page map is sent. The CE will then request the required LUT pages from Common Memory (the Intecolor, via IDOS-0), and send a message back to IDOS-1 when the load process is complete.

If this message is invalid, either because of its source (not a signal processing task), or because processing has not begun (the 'PPP' command was never entered), the message is displayed as a normal single block message.

5.3.2.2.4.1.3 Task Directive Reply (word 0 = 2)

This message signifies that a CE has successfully loaded a task as previously instructed by IDOS-1, and that it is awaiting further orders. If the task load operation was never requested by IDOS-1, the CE will be reset and reconfiguration invoked. If signal processing has not yet begun (via a 'PPP' command), the message is displayed as a normal single block message.

5.3.2.2.4.2 Status Reply Processing

When a status return is received, the first check is for the validity of the message. If processing has not yet begun (via a 'PPP' command), the message is displayed on the screen as if it were a single block message (message code 3). If processing is in progress, but the status was never requested by a previous message from IDOS-1, the card which sent the message is reset and reconfiguration is invoked.

Once the validity of the message is confirmed, the next check made is whether or not the card is a new addition to the system, or one which previously existed. New CEs are queued as idle and sent the system configuration table for future use. New IOCs are reset to ensure that they are in a known state, and queued as idle. CMs are queued, but are otherwise ignored.

Existing cards are checked for changes in status from previous reports. If an error has been detected, it is checked to see if it is fatal. Nonfatal errors are logged, but otherwise ignored. Fatal errors cause reconfiguration to be invoked.

5.3.2.2.4.3 Fault Message Processing

Fault messages are error reports that are serious enough to warrant immediate attention. They usually are accompanied by one or two data words to clarify the actual cause of the fault. These fault messages are translated by IDOS-1 into text strings which are displayed on the screen along with their associated data words. Red messages are fatal, and reconfiguration is invoked after the message is displayed. Yellow messages are nonfatal, and no further action is taken. (At present, all fault messages, fatal and nonfatal, are treated by IDOS-1 as fatal, and displayed in red.)

5.3.2.2.5 System Reconfiguration

In the event of a system failure such as a faulty CE or IOC, etc, IDOS-1 invokes the reconfiguration routine (RECNE). The purpose of this subroutine is to remove the faulty card and replace it if possible with a healthy spare in the system. The system status and configuration tables are automatically updated during reconfiguration.

The first item of business is to suspend all the tasks in the system to prevent spurious messages and lost data synchronization. Also involved in task suspension is the resetting of all IOCs (thus effectively suspending them also). Once all tasks have been suspended, the faulty card is reset by an executive message (message code 13), and its task placed in the task queue. The SYSTBL and STNDX are updated to show the failure before the routine terminates.

The actual process of re-loading the task into another card is performed at the next clock interrupt, if any spares are available.

5.4 Pulse-Pair Application Program (FTWRP)

The heart of the Fault-Tolerant Weather Radar Processor system is the signal processing applications task which runs in the Common Elements. This task has the identification 60 (octal), and executes in CEs with virtual addresses 60 through 64 (octal). The following sections describe the FTWRP program in detail, and a list of data structures appears in Appendix B. The processing flow diagram of Figure 2-4 provides a convenient reference for the descriptions that follow.

5.4.1 Overview of FTWRP Processing

Processing in FTWRP is divided among the CEs according to range. In the normal system, five CEs take equal shares of data, according to the total number of range cells being processed. For example, in a 1024 range cell application, the processing is divided up as follows:

<u>VA</u>	<u>Range Cells</u>
60	0 - 207
61	208 - 411
62	412 - 615
63	616 - 819
64	820 - 1023

5.4.1.1 Data Buffering in FTWRP

Processing dwells are triple buffered in the FTWRP applications software. That is, while processing is taking place out of one data buffer, another buffer is being filled with new data. Three buffers are used instead of two to take care of peak system load, and to prevent loss of data due to transient problems in CEs (e.g., I/O problems such as busy receivers).

Each time a new block of range cell data is received, a "pulse-level" interrupt routine is invoked, which performs the required operations (either autocorrelation and integration or simple integration), depending on whether the data is reflectivity or coherent channel information. The integration is accomplished by adding the result to the contents of the current input buffer. Once the buffer is full (a full complement of pulses have been integrated as determined by the NSI parameter), it is released for "dwell-level" processing which computes mean velocity, range normalization, shear, etc. Meanwhile, a new buffer is allocated to input, and pulse-level processing can continue.

5.4.2 FTWRP Functional Description

This section discusses the functional aspects of the FTWRP applications software. The Continuous Pulse Sequence processor is described here. The Dual Wavelength Sequence processor, which performs all the functions of the Continuous Pulse processor plus some complicated ambiguity resolution techniques, is discussed in Section 5.4.3.

5.4.2.1 Task Startup/Initialization

This section describes the process of bringing up the FTWRP applications task in a CE and preparing it to run. This involves task startup, processing startup and initialization, parameter loading, and look-up-table loading.

5.4.2.1.1 Task Startup

When IDOS-1 starts the task, FTWRP immediately enters an idle loop waiting for a processing startup command. This enables IDOS-1 to send commands to FTWRP without interfering with data processing, and without fear of destroying any packets or tables. Usually the commands from IDOS-1 include loading of

parameters and/or look-up tables. The formats for each of the possible commands which are honored by the applications software are detailed in Section 5.4.1.3.

5.4.2.1.2 Processing Startup/Initialization

When IDOS-1 finally sends the processing startup command to FTWRP, the first thing that is done is to initialize all the special instruction packets. By initializing once at the beginning rather than before each use of the instructions, the loops are made more efficient and problems caused by slow processors are minimized.

Once initialization is complete, the dwell processor enters a second idle loop waiting for an input buffer to be freed by the pulse-level routines. If this takes too long, a nonfatal error bit is set in the CE task status word.

5.4.2.1.3 Parameter Loading

Parameters may be changed at any time, either before processor startup, or during processing. It is recommended, however, that the processor be stopped before parameter lists be sent.

Once parameters have been changed or initialized according to the message from IDOS-1, the processor run flag is checked to see if the dwell-level processor is active. If so, the task initialization routine is re-executed to bring all tables up to date. If not, nothing is done, under the assumption that IDOS-1 will be starting the system again at some point in the future. When it does, the task will be re-initialized automatically.

5.4.2.1.4 Look-Up Table Loading

Look up tables are loaded by giving the CE a load map page number specifying how the look up table is to be loaded. The CE then requests the load map page from CM (the Intecolor), and uses it to determine which pages must be loaded next. The format of the Look-Up Table (LUT) page map is shown in Figure 5-10. When all the pages of the LUT have been successfully loaded, the CE sends a "LUT load complete" message to IDOS-1. (Section 5.3.2.2.4.1.2).

WORD	
0	PAGE COUNT (n)
1	PAGE NUMBER (0)
2	WORD COUNT (0)
3	PAGE NUMBER (1)
4	WORD COUNT (1)
⋮	⋮
2n-1	PAGE NUMBER (n-1)
2n	WORD COUNT (n-1)

Figure 5-10. Look-up Table Load Map Page Format

5.4.2.2 FTWRP Commands From IDOS-1

The FTWRP applications software recognizes six different commands from IDOS-1. Each command is a self contained message from virtual address 77 (octal) and with message code 5. The format for each command is shown in Table 5-13 and described in the following sections. Word 0 of the message body always contains the command code to be executed.

5.4.2.2.1 Change Virtual Address (Command 0)

This command is used only during spare rotation (see Section 5.6), and its purpose is to inform the CE that it will be asked to change its virtual address to the specified value as soon as the current pulse-level input buffers are filled.

Word 1 of the message contains the new virtual address.

5.4.2.2.2 Change Parameter(s) (Command 1)

This message instructs the program that a new batch of parameters are to be used in future processing. Once the new parameters have been put in place, the task initialization routine is called again.

Although parameters can be changed at any time, even during processing, it is not recommended without first stopping the signal processor with a command 4 (Section 5.4.1.3.4). If parameters are changed during processing, some packets may be temporarily destroyed, which will ultimately result in either bad data or the CE actually being reset due to a perceived fault. Therefore, it is best to stop the processor, load new parameters, and then restart it using command 5 (Section 5.4.2.2.6).

5.4.2.2.3 Change Look-Up Table Number 1 (Command 2)

This message instructs the processor to load the range normalization look-up table. Word 1 contains the number of the load map page which will supply the required information to load the table. The format of the load map page is shown in Figure 5-10.

Once the Look Up Table has been successfully loaded from Common Memory (the Intecolor), a message to that effect is transmitted to IDOS-1. The format of the load complete message has message code 3 and a 1 in word 0.

Table 5-13. Signal Processor Command Formats

<u>Word 0</u>	<u>Function</u>	<u>Parameters</u>
0	Change Virtual Address	Word 1 = new virtual address
1	Change SP Parameters	Word 1 = number of parameters (n) 2 = parameter number . . 2n Parameter number 2n+1 Parameter value
2	Load Range Normalization Look-up Table	Word 1 = Load map page number
3	Load Tangential Shear Look-up Table	Word 1 = Load map page number
4	Stop processing	
5	Start (Restart) processing	

5.4.2.2.4 Change Look-Up Table Number 2 (Command 3)

This message instructs the processor to load the Tangential Shear look-up table from Common Memory (the Intecolor). The contents of this table are the values corresponding to the tangential velocity of the beam versus the range cell number. The method and termination of loading the table are the same as discussed in Section 5.4.2.1.4.

5.4.2.2.5 Suspend Processing (Command 4)

This message instructs the program to halt signal processing after the current dwell of data is processed. New buffers of data may still be processed by the pulse-level routines, but are not passed to the dwell-level. The dwell processor re-enters the initialization idle loop, and will not exit until a new Start Processing command is received. There are no data words associated with this message.

5.4.2.2.6 Start Processing (Command 5)

This command is used to start the dwell-level processing from the initial conditions, or resume after a Suspend Processing command is received. The task initialization routine is invoked immediately after receipt of this message, and the dwell processor is entered. There are no parameter data words associated with this message.

5.4.2.3 Pulse-Level Processing

As stated above, the pulse-level processing consists of two parts: the coherent channel and reflectivity channel. These routines are discussed separately in the following sections.

5.4.2.3.1 Coherent Channel Processing

The coherent channel processing is performed in the interrupt handler IQPULS each time a new packet with message code 10 is received. The format of the coherent data packets is shown in Figure 4-12.

The heart of the coherent channel processing is the RACOR signal processing instruction which was developed for FTWRP. This instruction, described in detail in Section 6.1.10, takes the complex conjugate of the packed inphase and quadrature data, and multiplies it with the packet data from the previous pulse, which was stored in a delay buffer in memory. The result is then integrated over the entire dwell by being added to a cumulative I and Q data set maintained in memory. The actual number of pulses integrated in this manner is determined by the NSI parameter. The current data set is then stored in the delay buffer for processing the next pulse when it comes in.

A pulse count is maintained by IQPULS to determine when the dwell is complete. When a buffer is full, it is flagged as such and released to the dwell processor for the rest of the algorithm implementation.

The pulse after a complete dwell is treated as a new dwell data set, and therefore (since there is no previous data to be autocorrelated with) the packet is simply read into the delay buffer with the SREADR (scatter read) instruction.

The second pulse of data is autocorrelated with the delay buffer as with the RACOR instruction, but instead of the results being integrated into memory, they are simply stored there, effectively clearing the contents from previous dwells. This is accomplished with the RACORI instruction (see Section 6.1.9).

If all three coherent channel buffers are full, the last one filled will be emptied and refilled, and a nonfatal fault bit set in the task status word.

5.4.2.3.2 Reflectivity Channel Processing

The reflectivity channel data processing is performed by the interrupt service routine LZIPULS, which is invoked each time a packet with message code 11 is received. The format of the reflectivity data packet is shown in Figure 4-12.

As in the case with coherent channel processing, reflectivity processing revolves around a single instruction, RACC (Read and Accumulate). This relatively simple instruction (described in detail in Section 6.1.8) takes the unpacked reflectivity data and adds it to a buffer in memory.

A separate pulse counter is maintained by LZPULS to determine when a dwell is complete. Once the buffer contains a complete dwell, it is flagged appropriately and released to the dwell processor.

The new buffer is initialized by storing the next pulse of data directly over the old data with the READ instruction.

If all three reflectivity buffers are full, LZPULS will re-appropriate the buffer which was filled last, and set a fault bit in the task status word.

5.4.2.4 Dwell Level Processing

Although the Dwell level processing represents a single large routine in FTWRP, it is functionally similar to the pulse-level processor in that it also has two distinct emphases (reflectivity and coherent channel). To facilitate understanding, these will be discussed separately. The special signal processing instructions which were designed for FTWRP and used in the dwell processor include SCALE, CVEC, VADD, VSUB, and SLINT, and are described in detail in Section 6.1.

5.4.2.4.1 Common Preprocessing

The FTWRP software was designed to provide maximum accuracy while minimizing the probability of overflow errors. Therefore, the algorithm was sized using worst case maximum data with NSI = 256 (maximum pulse integration). However, when NSI is reduced, proper scaling must be done to prevent loss of accuracy. Thus, the first operation performed on all data is scaling so that as many significant bits as possible will be retained.

5.4.2.4.2 Reflectivity Processing (Dwell Level)

Reflectivity processing is relatively simple in comparison to that for the coherent channel. The major operation which must be performed is range normalization to account for signal attenuation with respect to range. This is accomplished by adding (using VADD) a constant which is determined by the range cell number. These constants are contained in a look-up table which is indexed by range cell number.

Range normalization may be turned off by the operator by typing

SET RNORM=OFF

which sets a flag in FTWRP that instructs the dwell processor to skip execution of the VADD instruction. typing

SET RNORM=ON

will turn the range normalization function on again. No other processing is performed on reflectivity data until the final common output processing.

5.4.2.4.3 Coherent Channel Processing (Dwell Level)

Coherent channel processing is somewhat more complicated, since a number of operations may need to be accomplished. The first, and most visible, function is the computation of the arctangent of the autocorrelation results from the pulse level routine. The result of the arctangent (computed by CVEC) is directly proportional to the mean velocity, since it is a measure of the Doppler phase shift of the returns. This information will be range-averaged and output to the Output Synchronizer as the mean velocity channel.

The second major function of the coherent channel processor is the computation of the shear. Either tangential or radial shear can be obtained, depending on the state of the SHEAR flag (set using the SET SHR command).

Radial shear is simplest, and consists of simply subtracting from each range cell the velocity of the previous range cell; that is,

$$\text{shear}(r) = \text{velocity}(r) - \text{velocity}(r-1)$$

where r is the range cell number.

Tangential shear is, simply stated, a comparison of the velocity from dwell to dwell; that is,

$$\text{shear}(r,n) = [\text{velocity}(r,n) - \text{velocity}(r,n-1)] * [1/(r*d\text{Theta})]$$

where r is the range cell number, n is the dwell number, and $d\text{Theta}$ is a measure of the angular velocity of the antenna in degrees per dwell. Therefore, the velocity information for each dwell must be stored in a delay buffer, to be subtracted from the new dwell when it comes in. The $1/(r*d\text{Theta})$ term is supplied by another look-up table indexed by range. After the tangential shear for a dwell is computed, the delay buffer is updated by copying the current dwell's velocity buffer into the delay buffer. This is accomplished most efficiently by using the SCALE instruction with a scale constant of one.

5.4.2.4.4 Common Post Processing

Once the desired data (reflectivity, mean velocity, and shear) has actually been computed, the results must be smoothed by sliding-window range averaging. This operation also serves to move the data into dedicated output buffers from which the messages to the output synchronizer can be built. The averaging process is performed by the SLINT instruction, unless the window size is set to 0 or 1 (effectively turning off the averaging function), in which case the faster SCALE instruction is used with a scale constant of 1.

It should be noted here that since the data has been partitioned among the CEs according to range, sliding window averaging becomes a bit of a problem around the range boundaries. At present, this is alleviated by each CE sending the next CE a block of data equal in size to the averaging window. This is not perfect, however, since CEs are not necessarily synchronized with each other; that is, one CE's dwell may be slightly skewed in time (a few pulses) from another. In the future, it may be found to be better to solve the problem by duplicating the first (or last) range cell in a block to fill out the average. Further experimentation is required in this area to arrive at final conclusions.

Once a CE has transmitted its few range cells to the next CE, it waits for a similar input from the CE which is processing the previous block. If it takes too long, it will go ahead and average, but set a fault bit in the task status word.

After averaging, the output packets must be built for transmission to the output synchronizer. The actual output process may involve as many as three packets of data (and thus, three output operations). The format of each of the packets is shown in Figure 4-16. The OSSA word is used by the output synchronizer to ensure that data is output to the Scan Converter in proper range order. The data words are merged together (three words per range cell) for up to 79 range cells of data per packet. Thus, in the case of NRC=256, only one packet is required per CE (maximum 52 range cells per CE), but when NRC=1024, three packets are necessary (two packets of 79 range cells, and one of 50).

After all the packets are output to the synchronizer, the dwell buffer is released back to the pulse-level processor, which may then refill it with a new dwell of data.

5.5 Dual Wavelength Application Task

The Dual Wavelength processing, depicted in Figure 2-5, is basically the same as for continuous pulse sequence processing, except for two extra processing modules labeled "Range Ambiguity Resolver" and "Coherent Channel Formatter." These software modules, called RAR and CCF respectively, help to map multiple trip echo ambiguities into their proper range intervals. This is accomplished by building a table (indexed by range) which specifies the "trip number" into which a range cell should be placed. The RAR module is responsible for creating the table, while CCF uses it to actually perform the mapping.

The reflectivity information consists of four independent blocks of range cells representing the same interval of each of four coherent channel trips. These four blocks are treated as a single large block during the normal pulse-level processing, since range dependencies do not exist there. The coherent channel is processed as before also, since there is at this point only one interval. In the dwell-level processing, the arctangent is computed without regard to range, but range normalization and shear computation are range dependent, and therefore the range ambiguity resolution must be performed first.

5.5.1 Range Ambiguity Resolver

A flowchart depicting the logic of RAR is shown in Figure 5-11. The first check performed by RAR is to determine the trip which contains the maximum reflectivity for a given range cell offset. At the same time, the relative difference between the maximum and the second largest trip must be computed. If this difference is less than a threshold (ZTH, set by the operator) the information is ambiguous, since the coherent channel information could conceivably be placed in either trip. In this case, an "ambiguous data" flag is placed in the RAR table. If one trip is clearly dominant, a trip "index" (0, 1, or 2) is entered into the RAR table. If the maximum reflectivity is below a threshold (PRETH, set by the operator) the reflectivity and coherent channels are all set to zero, and zero is entered into the RAR table. This process is repeated until all the coherent range cells have been identified with a trip index, or been found ambiguous.

NOTES: e = Reflectivity difference
 E = Index of trip in which max power is located
 A = Maximum power (reflectivity)
 z = reflectivity (range, trip)
 x = trip index
 r = range cell number (within trip)
 TABLE = Range Ambiguity Table (range)

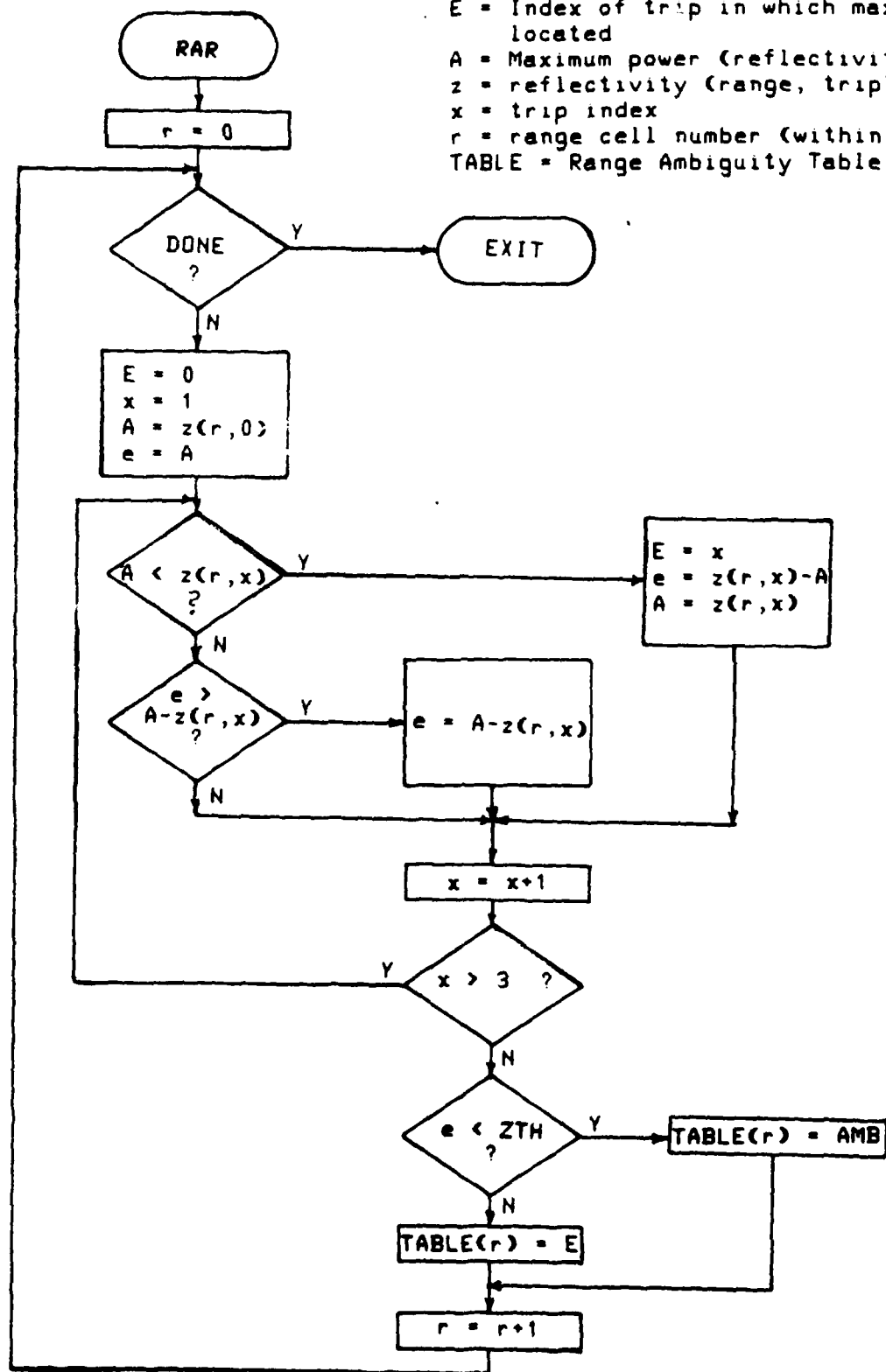


Figure 5-11. Range Ambiguity Resolver Flow Diagram

5.5.2 Coherent Channel Formatter

The CCF logic is shown in the flowchart of Figure 5-12. If the RAR table shows that the trip information is ambiguous, the first trip (trip 0) is assumed as the correct one, and the trip 3 range offset is given a code denoting that ambiguous data is being reported. The ambiguous data code is defined as (-1) times the block integration (BLINT) window size BLW, set by the operator) for reasons which will become clear later. All other trips (1 and 2) are set to zero. If nonambiguous data is encountered (i.e., the RAR table entry is 0, 1, or 2) the table entry is used to determine in which trip the coherent channel data should be placed (trip 3 is always set to zero, unless ambiguous data is encountered).

Once the entire RAR table has been processed, the coherent channel buffers will have expanded by a factor of four. Since range dependencies have presumably been resolved at this point, range normalization and shear calculations can be performed.

Since the Scan Converter (SCRM) cannot accept the expanded buffers of data, it must be reduced using the block integration instruction (BLINT). Once BLINT is performed, the reduced buffers are of the proper size for output to the existing hardware. Ambiguity information is retained, since the fourth trip (trip 3) still contains the ambiguous flags. However, since data reduction has taken place, resolution of which range cells were ambiguous may be lost. Since the ambiguous flag was defined as (-1) times the BLINT window size, the output of BLINT in trip 3 should be 0 (if no ambiguities existed in the window), or (-1) times the number of ambiguities encountered in the window. For example, if the BLINT window is 4 range cells, and two ambiguities are encountered within a window, the trip 3 output should be -2. Therefore, maximum ambiguity information is made available for post-processing.

NOTES: r = Range cell Number
 TABLE = RAR index table (range)
 VEL = Mean velocity output (range, trip)
 v = Mean velocity input (range)
 SHEAR = shear output (range, trip)
 s = shear input (range)

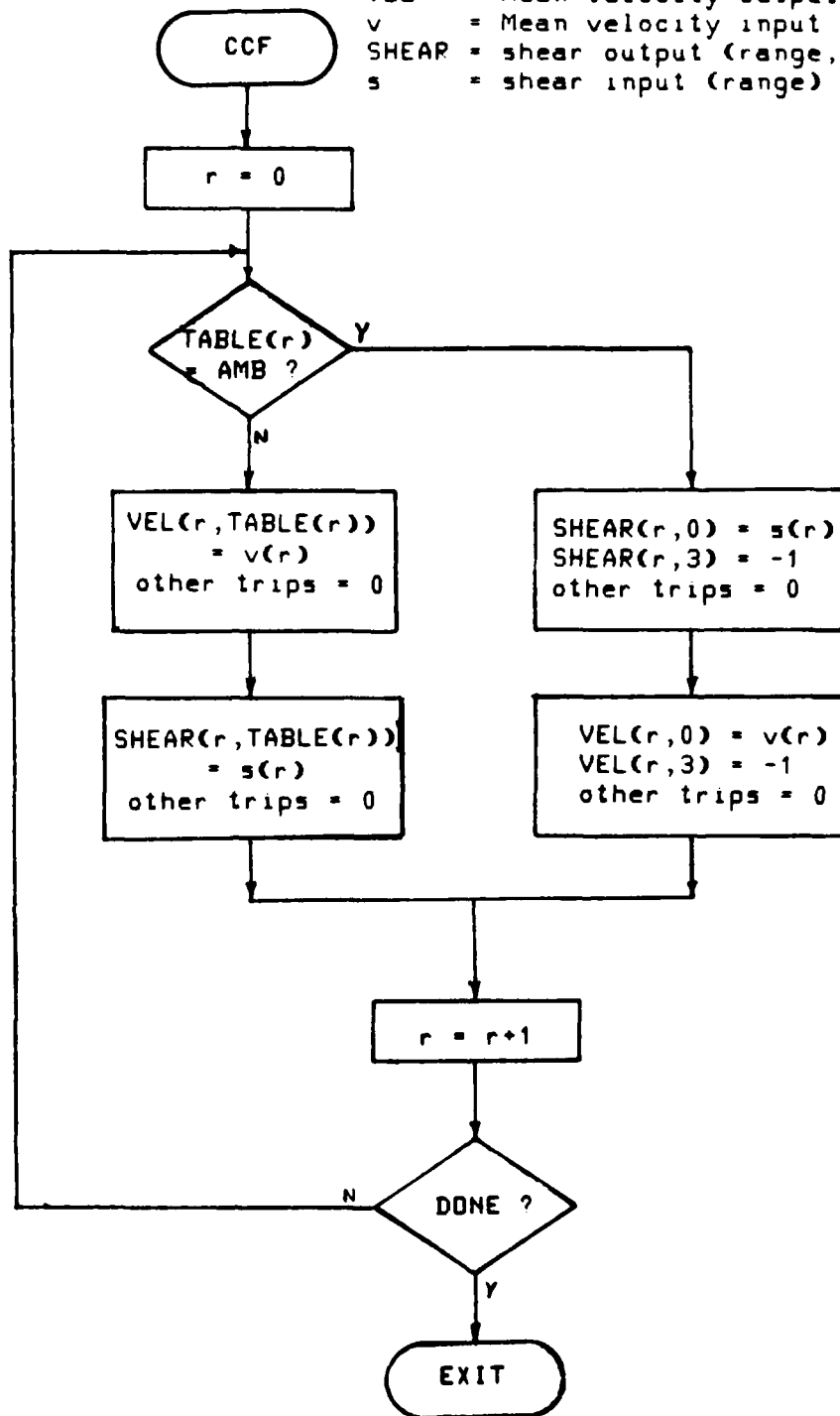


Figure 5-12. Coherent Channel Formatter Flow Diagram

5.6 Spare Rotation

One of the requirements of the Fault Tolerant Weather Radar Processor is the ability to run comprehensive diagnostics in individual CE's while the system is operating. Since these diagnostics cannot run concurrently with tasks in an active CE, a separate diagnostic task which executes in a "spare" (i.e., not used in the system at the time) CE is a good alternative.

If it is possible to "rotate" this task through different CE's in the system without losing data or computing power, all CE's may test themselves periodically to insure system integrity. This section describes the complications of such an operation, and outlines the scheme which satisfies the requirements.

Requirements

The requirements associated with spare rotation are:

- o No data may be lost
- o DOS-1 must initiate and control the operation
- o Unaffected CEs, IOCs, and CMs must not be cognizant of any change taking place.

Although these requirements seem obvious, they are by no means trivial to implement in FTWRP. To demonstrate this, the applicable environmental considerations of FTWRP are repeated in the following subsection.

5.6.1 FTWRP Spare Rotation Environment

5.6.1.1 The system consists of 5 processing CEs and one spare; two IOCs (one input, one output), and an Intecolor, in which runs IDOS-1; there are no CMs.

5.6.1.2 Processing is distributed among CEs according to range interval (e.g., CE #1 receives range cells 0-207, CE #2 receives cells 208 - 411, etc.), and all tasks are identical. However, constants and look-up tables are *range dependent*, and are therefore different in each CE.

5.6.1.3 Data is input via an IOC in continuous input mode, and therefore is received by CEs periodically without request. Since the IOC has a predefined sequence of virtual addresses (VA) to which to send data any CE

entering or leaving the processing stream must change its VA during a finite window of time to prevent loss of data. Therefore, the IOC gates the maximum time allowed for rotation, which may vary from 0.25 to 2.0 ms in length.

5.6.1.4 A block of radar data is input to each CE every 256 - 2048 μ s, upon which "pulse-level processing" (PLP) is performed. The results are integrated into dwells of 32 - 256 pulses each, upon which the CE performs "dwell-level processing" (DLP) before the data is output. While DLP is going on, the PLP of the next dwell takes place. Therefore, at any point in time two dwells of data are undergoing processing. From a spare rotation point of view, this means that the rotation process may only take place between the last pulse of the previous dwell and the first pulse of the next dwell. This window, 0.25 - 2.0 ms long occurs about 3 - 4% of the time. To complicate the picture even more, this window shifts in time from CE to CE and from dwell to dwell.

5.6.1.5 The time required to load a new task into a CE is on the order of 20 - 30 seconds, and essentially halts the execution of IDOS-1. This requires the spare to load the application task in advance of the actual rotation process.

5.6.2 Spare Rotation Implementation

The small window size and lack of synchronization discussed in the previous section makes it extremely difficult for IDOS-1 to control rotation, since IDOS-1 never really knows a CE's progress in the processing sequence. However, with the added capability of DOS-0 to change its own virtual address with previous authorization from IDOS-1 (see system request 9, section 5.1.4.8) the following scheme became possible:

1. Rotation starts with CEs #1 - 5 doing signal processing, while CE #6 performs diagnostics.
2. CE #6 finishes its task, and reports no errors to IDOS-1.
3. IDOS-1 determines the next CE to run diagnostics (CE #1)
4. IDOS-1 instructs CE #6 to load the signal processing task with CE #1's tables, and begin execution under a pre-defined unused virtual address. CE #6 is now waiting for data to work-on, but receives

none because of its address.

5. After CE #6 has started, IDOS-1 instructs CE #1 to change its virtual address at its earliest convenience.
6. When CE #1 comes to the appropriate time to rotate, it informs CE #6 to change its VA to that of CE #1, and immediately changes its own to that of a spare.
7. CE #6 now starts receiving input and performs PLP and DLP in the normal manner.
8. CE #1 continues DLP on its data, then informs IDOS-1 that input has ceased (a fault message).
9. IDOS-1, expecting the message from CE #1, informs it to load and execute the diagnostic task.
10. During steps 3 through 9, no status polling may be done.
11. This process is repeated until all CEs have run the diagnostics.

This procedure, while keeping IDOS-1 in ultimate control, gives individual CEs enough autonomy to synchronize rotation with the data. IDOS-1 is aware of all that goes on, but other CEs and the IOCs are unaffected. Most important, no data is lost.

REFERENCES

- 5-1. M.J. Young memo, "Intecolor-CE Communication", dated 8 February 1979, MJY-048, EM78-0104B.

6. FIRMWARE DESCRIPTION

This section describes firmware developed specifically for the FTWRP system, including the special CE instructions and IOC programs. Previously existing firmware, or firmware not developed uniquely for FTWRP will not be discussed in the document, but may be found in Reference 4-2.

6.1 New Common Element Microcode

Ten new instructions were added to the CE repertoire to meet the processing requirements of the FTWRP. They represent an expansion of the existing capabilities of the CE as described in the FTSP Programmer's Handbook (Reference 4-2).

All ten new instructions were designed to be interruptable in the midst of execution. If an interrupt occurs, all internal working registers are pushed on the user stack, and the interrupt is serviced. Upon completion of the interrupt service routine, the registers are popped off the stack and the instruction is restarted from the point of interruption. Any interrupt may be honored, including refresh, memory protect violations, and I/O complete. Trace is not a valid interrupt during execution of an instruction.

It should be noted here that the instructions described below do not have parameter error checking capabilities. Should an erroneous parameter be issued, the instruction may produce undefined results.

6.1.1 Vector Add (VADD)

Each two's complement 16-bit number in the first input buffer (b1) is added to the corresponding number of the second input buffer (b2), and the 16-bit sums are placed sequentially in the output buffer. The overflow flag is set if any two's complement overflow is detected during execution.

Parameters are passed in a contiguous block of memory at the address given in the A register:

<u>Address</u>	<u>Argument</u>
(A)	Starting Address of Output Buffer
(A)+1	Starting Address of Input Buffer (b1)
(A)+2	Starting Address of Input Buffer (b2)
(A)+3	Number of words to be processed ($0 < n < 65536$)

Execution time: $14 + 6n$ cycles.

6.1.2 Vector Subtract (VSUB)

Each two's complement 16-bit number in the second input buffer (b2) is subtracted from the corresponding number in the first input buffer (b1), and the differences are placed sequentially in the output buffer. The overflow flag is set if any two's complement overflow is detected during execution.

Parameters are passed in a contiguous block of memory at the address given in the A register:

<u>Address</u>	<u>Argument</u>
(A)	Starting Address of Output Buffer
(A)+1	Starting Address of Input Buffer (b1)
(A)+2	Starting Address of Input Buffer (b2)
(A)+3	Number of words to be processed ($0 < n < 65536$)

Execution time: $14 + 6n$ cycles.

6.1.3 Vector Multiply (VMULT)

Each two's complement 16-bit number in the second input buffer (b2) is multiplied by the corresponding number in the first input buffer (b1), and the sixteen most significant bits of the 32-bit product are placed sequentially in the output buffer. The overflow flag is unaffected by this instruction.

Parameters are passed in a contiguous block of memory at the address given in the A register:

<u>Address</u>	<u>Argument</u>
(A)	Starting Address of Output Buffer
(A)+1	Starting Address of Input Buffer (b1)
(A)+2	Starting Address of Input Buffer (b2)
(A)+3	Number of words to be processed ($0 < n < 65536$)

Execution time: $16 + 6n$ cycles.

6.1.4 Block Integration (BLINT)

This instruction is used in data reduction in the dual-wavelength mode (not yet supported), by averaging a block of numbers and outputting a single result.

Block integration is performed in the following way: the input buffer is divided into blocks of length m . The average of each block is then computed by taking the sum of the quotients obtained by dividing each 16-bit number in

the block by the block length m . The results are then stored sequentially in the output buffer. The length of the output buffer (n) is thus the length of the input buffer divided by the block length. The overflow flag is unaffected by this instruction.

Parameters are passed in a contiguous block of memory at the address given in the A register:

<u>Address</u>	<u>Argument</u>
(A)	Starting Address of Output Buffer
(A)+1	Starting Address of Input Buffer
(A)+2	Length of Block ($m > 1$)
(A)+3	Length of Output Buffer ($n > 0$)

Execution time: $35 + (2m + 3)n$ cycles.

6.1.5 Sliding Window Integration (SLINT)

This instruction is used in FTWRP as a "smoothing" function just before the data is output to the SCRM display.

Sliding window averaging is performed in the following way: beginning with the first element in the input buffer, one window (length m) of data is isolated. The average of this window is then computed by taking the sum of the quotients obtained by dividing each element by the length of the window (m). The window is then "slid" by one element and the new window average is computed by subtracting (from the previous average) the first element quotient and adding the quotient from the last element of the new window. This is repeated until the last element of the input buffer becomes a member of the window. Averages for each window position are placed sequentially in the output buffer, which will have length $n-m+1$, where n is the input buffer length.

The parameters are passed in a contiguous block of memory at the address given in the A register:

<u>Address</u>	<u>Argument</u>
(A)	Starting Address of Output Buffer
(A)+1	Starting Address of Input Buffer
(A)+2	Window Length ($m > 1$)
(A)+3	Input Buffer Length ($n > 0$)

Execution time: $39 + 2m + 5(n-m)$ cycles.

6.1.6 Scale (SCALE)

As the name implies, this instruction is used in FTWRP to scale the pulse-level results before dwell-level computation is performed. The purpose of this scaling is to preserve accuracy while preventing overflow errors during computation. SCALE also has a feature which permits an address increment (m) to be included, so that only every " m 'th" data element is scaled.

The two's complement 16-bit number in every m 'th element of the input buffer is multiplied by the scaling constant. The 16 least significant bits of the 32-bit product are stored in every m 'th location of the output buffer. The overflow flag is unaffected by this operation.

Parameters are passed in a contiguous block of memory at the address given in the A register:

<u>Address</u>	<u>Argument</u>
(A)	Starting Address of Output Buffer
(A)+1	Scaling Constant
(A)+2	Starting Address of Input Buffer
(A)+3	Address Increment ($m > 0$)
(A)+4	Number of words to be processed ($n > 0$)

Execution time: $16 + 4n$ cycles.

6.1.7 Circular Vectoring (CVEC)

This instruction takes the vector in rectangular (x,y) coordinates and converts it to polar (magnitude,angle) coordinates. It is used in FTWRP to compute the average phase shift of the coherent returns, which is proportional to the mean velocity.

Starting with the first location of the input buffer, the two-word block beginning at every i 'th location (where i is the address increment) is taken as an x,y two's complement coordinate pair, of 16 bits each. For each pair, the magnitude and angle equivalents are computed and stored sequentially in the appropriate output buffers. The angle is returned in two's complement form with a range of $+179.99450$ to -180 degrees scaled over the available 16 bits. The instruction accepts vectors in all four quadrants.

The algorithm used is an iterative process which increases in accuracy with the number of iterations performed. Angle accuracy is approximately one bit per iteration. Empirical observation shows that maximum angular accuracy is obtained with about 13 iterations, while maximum magnitude accuracy requires only about 8. Table 6-1 provides accuracy information for each iteration. For more information on the CVEC algorithm, see Appendix C.

The magnitude outputs are automatically scaled by approximately the square-root of 2 to prevent overflow errors.

Parameters are passed in a contiguous block of memory at the address given in the A register:

<u>Address</u>	<u>Argument</u>
(A)	Starting Address of Angle Output Buffer
(A)+1	Starting Address of Magnitude Output Buffer
(A)+2	Starting Address of Input Buffer
(A)+3	Input Buffer Address Increment ($i > 1$)
(A)+4	Number of x,y Pairs to be Processed ($n > 0$)
(A)+5	Number of Iterations ($0 \leq m \leq 13$)

Execution time: $23 + m + (7m + 26)n$ cycles.

6.1.8 Read and Accumulate (RACC)

This instruction takes the 16-bit numbers in the I/O receiver buffer and accumulates them into a sum buffer in memory, and it is used in FTWRP as the means by which reflectivity returns are integrated into dwell buffers. RACC has the ability to extract a non-data "message" from the top of the I/O buffer before processing begins. In FTWRP, this "message" (2 words) contains information from the Input Synchronizer about the pulse number, block number, pulse width, and pulse repetition interval.

On entry, RACC assumes that all header words have been removed from the receiver buffer, and that the next word read will be the packet word count. This word count will be thrown away. The next m words will be extracted and placed sequentially in memory starting at the specified message buffer address. Then, each two's complement 16-bit number in the output buffer is replaced by the sum of itself and the corresponding data word in the receiver buffer. The overflow flag is set if any two's complement overflow is detected during execution. On completion, the CE receiver is reset.

Table 6-1. CVEC Accuracy

<u>Iteration</u>	<u>Angular Accuracy</u>	<u>Magnitude Accuracy</u>
0	$\pm 45^{\circ}$	+ 0%, -29.29%
1	$\pm 22.5^{\circ}$	+ 0%, -7.61%
2	$\pm 11.25^{\circ}$	+ 0%, -1.92%
3	$\pm 5.63^{\circ}$	+ 0%, -0.482%
4	$\pm 2.81^{\circ}$	+ 0%, -0.120%
5	$\pm 1.41^{\circ}$	+ 0%, -0.0301%
6	$\pm .703^{\circ}$	+ 0%, -0.00753%
7	$\pm .352^{\circ}$	+ 0%, -0.00188%
8	$\pm .176^{\circ}$	+ 0%, -<0.0015%
9	$\pm .0879^{\circ}$	+ 0%, -<0.0015%
10	$\pm .0439^{\circ}$	+ 0%, -<0.0015%
11	$\pm .0220^{\circ}$	+ 0%, -<0.0015%
12	$\pm .0110^{\circ}$	+ 0%, -<0.0015%
13	$\pm .00550^{\circ}$	+ 0%, -<0.0015%

Because RACC operates directly out of the I/O receiver buffer, it is a privileged instruction, that is, it may only be executed while the CE is in the privileged mode. Execution while in non-privileged mode will generate an illegal instruction exception.

Parameters are passed in a contiguous block of memory at the address given in the A register:

<u>Address</u>	<u>Argument</u>
(A)	Starting Address of Message Buffer
(A)+1	Length of message minus 1 ($m-1$) ($m > 1$)
(A)+2	Starting Address of Output Buffer
(A)+3	Length of Output Buffer ($n > 0$)

Execution time: $16 + m + 3n$ cycles.

6.1.9 Read and Autocorrelate Initial Results (RACORI)

This instruction, along with the next (RACOR) are used in FTWRP to compute the autocorrelation function of the coherent channel returns, and to integrate them into a dwell buffer. RACORI is different from RACOR only in that the results are stored in the output buffer rather than added to it.

It is assumed that the output buffer will already contain the packed data of the previous pulse, spread out by three locations; that is, there must be 2 empty locations after each packed data word. The format of the packed data is as shown in Figure 6-1.

On entry, the receiver buffer is assumed to contain a word count (thrown away), followed by m message words (as in the RACC instruction), and then n words of packed inphase (I) and quadrature (Q) data. The message words are first extracted and placed in the message buffer according to the parameters. The complex conjugate of each packed data word is then multiplied (complex multiplication) by the packed data in the corresponding location of the output buffer. The 8 most significant bits from 16-bit I and Q products are then sign-extended and stored sequentially in the two locations immediately following the packed data in memory. The packed data in memory is then replaced by the one in the receiver buffer. The overflow flag is set if any two's complement overflow is detected during execution. On completion, the receiver is reset.

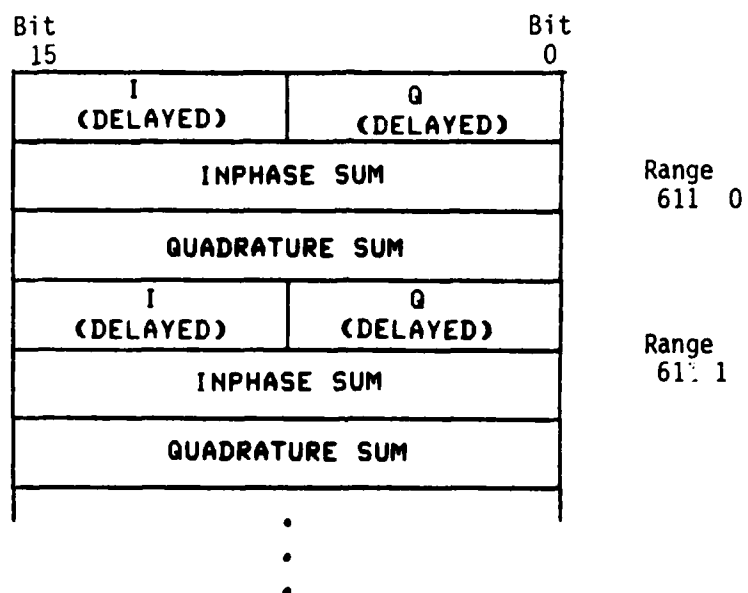


Figure 6-1. RACOR and RACORI Data Format

Because RACORI operates directly out of the I/O receiver buffer, it is a privileged instruction. Execution of RACORI in the non-privileged mode will generate an illegal instruction exception.

Parameters are passed in a contiguous block of memory at the address given in the A register:

<u>Address</u>	<u>Argument</u>
(A)	Starting Address of Message Buffer
(A)+1	Message Length minus 1 (m-1) (m > 1)
(A)+2	Starting Address of Output Buffer
(A)+3	Number of Packed Data Words Processed (n > 0)

Execution time: $17 + m + 9n$ cycles.

6.1.10 Read and Autocorrelate (RACOR)

This instruction is identical to RACORI except that the resultant I and Q products are added to the contents of the output buffer rather than stored. All assumptions and constraints are the same as for RACORI, as are the parameters lists.

As is the case with RACORI, RACOR is a privileged instruction, since it operates directly out of the receiver buffer. Execution of this instruction in the non-privileged mode will generate an illegal instruction exception.

Execution time: $17 + m + 11n$ cycles.

6.2 IOC Firmware Development

Ordinarily, the IOC supports only a standard "dynamic" input and output to peripherals, in which a CE must explicitly request input before the IOC will respond. However, due to severe time constraints and high DOS-0 overhead associated with I/O, the need for a new, "continuous input" mode in the IOC became apparent. This mode, designed specifically for the FTWRP, is used to distribute coherent and reflectivity channel data from the Input Synchronizer to the respective processing CEs. This section describes the Continuous Input Mode firmware which was developed for the 8x300-based IOC, and the IOC hardware changes which were needed to accommodate the new mode. Appendix B contains a table of major subroutines used by the Continuous Input Mode, and their functions.

The major difference in the Continuous Input Mode is that the IOC now becomes an active member of the system, rather than merely passive. That is, where before the IOC only responded to requests from CEs, it now spontaneously sends unsolicited messages to everyone.

6.2.1 Data Flow Overview

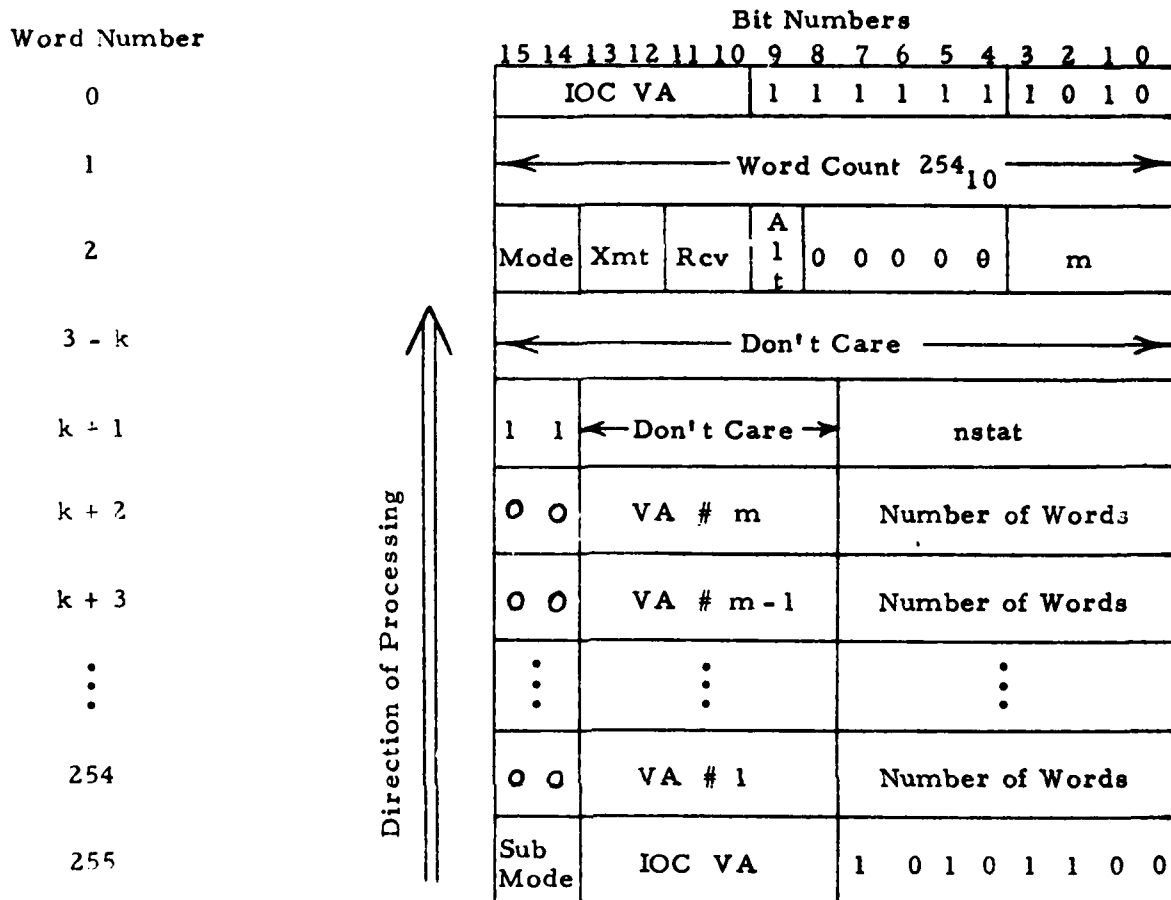
Figure 4-13 demonstrates the basic flow of data in the Continuous Input Mode. The first line displays the range blocks into which the data is conceptually divided (As mentioned earlier, radar returns are distributed among the CEs according to range, as shown in Figure 4-11). The second and third lines show the active time during which the IOC is attempting to extract data from the input synchronizer (by activating the select lines). And finally, the last line in the timing diagram shows the time during which the CE busses are active, when data packets are being shipped to the respective processing CEs (numbers depict the range block to which the data packet belongs).

In the normal system, there are 5 processing CEs, and thus 5 range "blocks", numbered in Figure 6-2 as 1 through 5. At system startup, the IOC must accept and distribute the first four blocks of coherent channel data before the first block of reflectivity data is accepted. This "skew" in data retrieval serves to spread out the data rates to each CE, and supply sufficient time for pulse-level computations to take place in the receiver buffers. Since coherent channel computations are inherently more complex than reflectivity integration, and since the CEs receiver is busy during the entire pulse-level computation, reflectivity packets must be delayed as long as possible. Analysis showed four-block skew in Figure 4-13 to be optimum.

Once the built-in time skew has been created, the IOC will begin to alternately accept and distribute coherent and reflectivity data among the CEs, as shown in Figure 4-13.

6.2.2 Exception Handling

In order to ensure proper operation of FTWRP in a degraded mode, certain specific requirements have been imposed on the method of exception handling. Once the IOC has been placed in the Continuous Input Mode, all inputs from the A and B busses (except RESET) will be ignored. This is accomplished



Mode = 01 For Continuous Input

Xmit = $\begin{cases} 0X & \text{To Xmit on Bus A} \\ 1X & \text{To Xmit on Bus B} \end{cases}$

Rcv. = $\begin{cases} 10 & \text{To Rcv. on Bus C, D Disabled} \\ 01 & \text{To Rcv. on Bus D, C Disabled} \end{cases}$

Sub Mode = 00 - AFGL Continuous Input

Alt = $\begin{cases} 0 & \text{No bus alternating in case of bus faults} \\ 1 & \text{Alternate A \& B buses for bus faults} \end{cases}$

m = Number of CE's to be sent data

nstat = Number of iterations between output of status message to DOS-1

Figure 6-2. Configuration Control Message for Continuous Input Mode

by setting the A and B bus receiver logic to the "busy" state. Therefore, no status polling by IDOS-1 is permitted, since the status request message will never be received. All fault conditions are assumed to be detected and flagged by the CEs, which in turn will inform IDOS-1 of the nature of the problem.

6.2.2.1 I/O Exceptions

When the IOC attempts to ship data to a CE and finds the receiver busy, it will immediately "drop" the packet on the floor by resetting the transmitter. The same holds true for NO REPLY, ARBITRATION FAULT, and TIMEOUT errors. The lost block will not be recovered, but will be replaced by the corresponding block from the next pulse.

The reasoning behind this action is that the IOC is under severe time constraints to get the data out of the Input Synchronizer and out to the CEs before the Input Synchronizer's buffers overflow. Appendix D has a more complete discussion of timing considerations in FTWRP, including worst-case loading of CEs and IOCs in various scenarios.

6.2.2.2 Input Synchronizer Overrun Faults

In the event that the Input Synchronizer's buffers overflow, it will signal the error to the IOC by activating the PARITY line. Both IOCs have been modified to bypass parity checking on the C-port, so that the PARITY line may be monitored directly. If the IOC detects the PARITY line becoming active, all operations cease, and the IOC enters an infinite loop. The CEs, when they stop receiving data, log the fault by setting status bits in their task status words. When IDOS-1 notices the lack of data in all the CEs, and determines that it is not an isolated CE problem, it will reset the IOC, reconfigure the system and attempt to restart the IOC.

6.2.3 IOC Continuous Input Mode Setup

The IOC is instructed to enter the Continuous Input Mode by a special form of the Configuration Control Message (message code 10). The new format is shown in Figure 6-2. The message contains a list (in reverse order) of virtual addresses to which the data is to be sent, along with the word counts

for each address. The end of the list is signified by bits 15 and 14 set in the mode field of the list. The list is stored in the uppermost locations of both of the I/O RAMs (X and Y RAMs), with the first address (block #1) located at RAM address 255, the second at 254, etc.

6.2.4 Continuous Input Mode Functional Description

The following sections describe the IOC Continuous Input Mode program logic as it relates to FTWRP.

6.2.4.1 Continuous Input Mode Initialization/Startup

Upon entry into the Continuous Input Mode, the RAM which received the configuration control message will already contain the list of CE virtual addresses and word counts in the form specified by Figure 6-2. Therefore, the first order of business is to copy this information into the other RAM so that both the X and Y RAMs will have the same data. This will save time and effort during execution by eliminating the concern of which RAM is available for access by the 8X300 and which one contains the list. Once both RAMs contain the entire header list, the next step is to initialize the Input Synchronizer by activating the number 2 select line (SEL2).

6.2.4.2 Initial Pulse Sequence

As mentioned above and shown in Figure 4-13, there is a built-in "skew" in the transfer of coherent channel and reflectivity data in the FTWRP. This four-block delay in the output of reflectivity is effected by a special sequence of code (the segment beginning with the label CMSTRT) which treats the first four blocks after initialization uniquely.

Each block of coherent channel data is extracted from the Input Synchronizer by activating the number 0 select line (SELO). Once a receiver interrupt (RINT) condition is sensed, signifying that the block has been transferred to the receiving RAM, the Receiver End (REND) status line is polled to determine whether the interrupt was legitimate (i.e., the block transfer was successful), or a fault condition (i.e., the PARITY line is active, signifying that the Input Synchronizer is overloaded). If REND is not active, the IOC will enter an infinite loop at the label INTC (a fault condition). Otherwise, the Synchronizer is unselected, and the block is prepared for transfer to the appropriate CE.

Before the block can be transmitted, however, the IOC must prepare the other RAM for input by building a new header word/word count in the first two words of the RAM. Internal IOC bus conflicts make it impossible to do this after the IOC-to-CE packet transfer begins. Once complete, however, the packet transmission may begin, and is effected by the subroutine CMTN.

The IOC then re-selects the Input Synchronizer using SEL0, and waits for the new data block to be successfully received. Once the packet is in, the transmitter status is checked to be certain that the previous packet has been successfully transferred to the CE. If it is not yet complete, the IOC waits for the transmitter interrupt (TINT).

The above process is repeated until four consecutive blocks of coherent channel data have been transferred to the appropriate CEs. At that time, the IOC enters the main loop of the Continuous Input Mode program.

6.2.4.3 Continuous Input Mode Main Processing Loop

The main loop of the Continuous Input Mode program works essentially the same as the initial loop, except that the decision of what gets input from the Input Synchronizer is more complex. The procedure INPSEL determines whether the Input Synchronizer should be selected for coherent data (SEL0) or reflectivity data (SEL1). The decision of which CE should receive the current block is performed by the procedure INPSEL as follows: if coherent data is input next, the address in internal register R5 is used to access the RAM to extract the next VA and word count. Otherwise, the address maintained in R1 is used. Throughout the Continuous Input Mode processing, these two registers contain separate values reflecting the current position in the virtual address list.

APPENDIX A

Intecolor/CE Support

- A-1 M. J. Young Memo "EPROM Programmer for Common Element," dated 20 December 1977, MJY-02, EM77-0570.
- A-2 FTWRP Special Instruction Microcode Test Program.
- A-3 FTWRP Test Target Generator.
- A-4 V. E. Follansbee Memo, "SEEK IGL00 Common Element Cross Assembler," dated 22 December 1977, VEF:77:14, EM77-0571.
- A-5 V. E. Follansbee Memo, "Common Element Cross Assembler Post Processor," dated 6 January 1978, VEF:78:01, EM78-0004.
- A-6 V. A. Jelich Memo, "Cyber/Intecolor Support Software," dated 6 January 1977, VAJ:78:01, EM78-0003.
- A-7 V. E. Follansbee Memo, "Intecolor Terminal Transfer Program Description," dated 10 January 1978, VEF:78:03, EM78-0013.
- A-8 R. J. Bonneau Memo, "MODDOS - Utility Program for Inspecting and Modifying DFTSP DOS-0 Object Code," dated 2 August 1978, RJB-124, EM78-0427.
- A-9 R. J. Bonneau Memo, "NEWMOD - MODDOS Enhancement," dated 30 October 1979, RJB-195, EM79-0632.
- A-10 M. J. Young Memo, "Modifying Files on the Intecolor," dated 26 January 1978, MJY-03.
- A-11 Intecolor Utility Routines.



FORM 10-0557 (9-65) 8090

DIVISION EQUIPMENT
Operation EDL
Department ADL - Wayland

To G. A. Works
From M. J. Young
Subject EPROM Programmer for
Common Element

Classification Unclassified
Contract No. 77D-209
Distribution DFTSP List
File No. EM77-0570
Memo No. MJY-02
Date 20 December 1977

The CE microcode will be stored in Intel 2716 UV Erasable PROMs. Since, at least during the debugging stages, many changes to the microcode will be necessary, a convenient in-house method of programming these PROMs is desired. If program tapes had to be sent to another location for programming each time a change was made, much time would be wasted.

The Intecolor 8051 which was purchased for use with the CE is ideal for this purpose. The assembled microcode can easily be stored on a floppy disk and edited using the terminal's editing commands. Since the 2716 EPROMs require only TTL level pulses during programming, a simple routine utilizing the 24-bit bi-directional I/O port would perform the necessary functions.

This memo describes the operation and use of a PROM programmer routine for Intel 2716's. Source listings of the BASIC program and the Assembly Language I/O driver are also included.

1. General Description

The PROM programmer was written with the assumption that a file of the proper format, containing the microcode is already resident on a floppy disk. A program which loads an assembled microcode file from Bedford into file slices 8 bits wide on floppy disks is being developed by Val Jelich, and should be available shortly.

A simple "personality board" which contains the necessary +5V and +25V power supplies is connected to the 24-bit I/O port in the rear of the terminal. To prevent possible damage to the PROMs, a switch is provided to turn the high-voltage supply on or off when instructed by the program.

The routine itself was designed to make programming the PROMs as simple as possible. Files of any length (less than 2048 bytes) may be programmed, or an "EDIT" mode may be used to selectively alter specific locations in the PROM. After programming is complete, the data is verified to insure that all locations were programmed properly. If an error occurs, a message is printed on the console, and the faulty data is displayed. The user may then enter (or reenter) the EDIT mode and attempt to correct the faulty location. Once the faulty location is corrected, the entire file can be verified again (without reprogramming) using the VERIFY option.

2. Operating the Programmer

The PROM programmer is stored on the Common Element floppy disk (volume name = CESYN1118) under the name EPROM.BAS. This disk should always be kept next to the terminal, along with the demo disk supplied by Bartlett Associates (volume name = ISCM100677). The I/O driver for the programmer is stored separately under the name EPROM.DAT and is loaded into the RAM by the main program.

To use the programmer:

1. Enter the BASIC Operating System --

a. To initialize BASIC, type ESCAPE then W.

The terminal will respond with MAXIMUM RAM ADDRESS?
For most purposes, a maximum address of 49151 is sufficient.
When this number is typed in, BASIC should respond with READY.

b. Once BASIC has been initialized, there is no need to do it again each time the operating system is entered. To reenter BASIC, type ESCAPE then E. If BASIC had been initialized previously, it will respond with READY. Otherwise, it must be initialized as shown in step (a).

2. Insert the Common Element diskette in drive 0 (left drive).

3. Type LOADPRINT "EPROM".

The terminal should respond with READY.

4. Type RUN.

The program will load the I/O driver into RAM, then clear the screen and type "Intel 2716 EPROM Programmer", followed by a set of operating instructions. These are listed here for completeness:

1. Plug the programmer board into the 24-bit I/O port in back of the Intecolor 8051.
2. Plug the board into an AC outlet. The power-on indicator should light.
3. To avoid possible damage to the PROMs, DO NOT TURN ON THE +25V SUPPLY UNTIL INSTRUCTED TO DO SO.

4. When the program responds with EDIT, FILE, or READ; type EDIT if single bytes are to be modified, FILE if an entire file is to be programmed, or READ if the PROM is to be read and displayed.
5. The file name must be appended with a .DAT extension (Example: PROM1.DAT).
6. If file processing is to be performed, the program responds with OPTION: after the PROM address has been input. The options are: PROGRAM (Program and Verify) and VERIFY (Verify only).
7. When prompted with commands like TURN ON +25V; or INSERT PROM IN SOCKET:, respond by typing any character followed by a carriage return. Just typing a carriage return will cause a program exit. To reenter the program at the point of exit, type CONT.

Once programming begins, the keyboard will lock until all the data has been processed. This will take approximately 2 1/2 minutes.. If for any reason programming must be halted prematurely, turn off the +25V supply, then press the CPU RESET button. This will place the terminal in the CRT mode, but BASIC may be reentered by typing ESCAPE E.

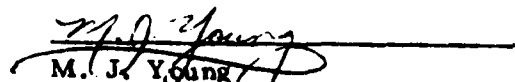
A sample programming session is included in Attachment 1. If there are any problems or questions, I can be reached at extension 2563.

MJY/ild

Attachments:

1. Sample Programming Session
2. BASIC Source Listing
3. I/O Driver

cc: Digital Fault-Tolerant Signal Processor List (attached)


M. J. Young
Advanced Electronic Techniques
Wayland Box M9, x2563

SAMPLE PROGRAMMING SESSION-12/2/77

PAGE 1

(DATA TYPED IN BY USER IS ENCLOSED IN BRACKETS [])

YOU ARE NOW IN THE CRT MODE
 [(ESCAPE)(W)]
 (ERASE SCREEN)
 INTECOLOR 8001 BASIC COPYRIGHT 1977 BY CHARLES F. NUENCH
 MAXIMUM RAM ADDRESS?[49151]
 READY
 [LOADPRINT*EPROM*]

READY
 [RUN]
 (ERASE SCREEN)

INTEL 2716 EPROM PROGRAMMER

INSTRUCTIONS

1. PLUG PROGRAMMER BOARD INTO THE 24-BIT I/O PORT IN BACK OF THE INTECOLOR 8051
2. PLUG THE BOARD INTO AN AC OUTLET. THE POWER ON INDICATOR SHOULD COME ON
3. DO NOT TURN ON THE +25V POWER UNTIL INSTRUCTED TO DO SO
4. WHEN THE PROGRAM PROMPTS WITH EDIT, FILE, OR READ:, TYPE EDIT IF SINGLE BYTES ARE TO BE MODIFIED, FILE IF AN ENTIRE FILE IS TO BE PROGRAMMED, OR READ IF THE PROM IS TO BE READ AND DISPLAYED.
5. THE FILENAME MUST BE APPENDED WITH A .DAT EXTENSION
6. IF A FILE IS TO BE PROCESSED, THE PROGRAM RESPONDS WITH OPTION: AFTER THE PROM ADDRESS HAS BEEN INPUT. THE OPTIONS ARE PROGRAM (PROGRAM AND VERIFY) AND VERIFY (VERIFY ONLY)
7. WHEN PROMPTED WITH COMMANDS LIKE INSERT PROM IN SOCKET: RESPOND BY TYPING ANY CHARACTER FOLLOWED BY A CARRIAGE RETURN. JUST TYPING A CARRIAGE RETURN WILL CAUSE A PROGRAM EXIT. TO RETURN TO THE PROGRAM AT THE POINT OF EXIT, TYPE CONT.

[EDIT OR FILE:[FILE]
 FILENAME:[PROM1.DAT]
 EPROM ADDRESS IN HEX:[0]
 DI: IUN:[PROGRAM]
 INSERT PROM IN SOCKET:[X]
 TURN ON +25V:[X]

VERIFY: ERROR
 DATA:5F
 FROM:4E
 ADDRESS AT WHICH ERROR OCCURRED=1A9

TURN OFF +25V:[X]

MORE (Y OR N):[Y]

[EDIT OR FILE:[EDIT]]
[FROM ADDRESS:[1A9]]
- [INSERT FROM IN SOCKET:[X]]
[TURN ON +25V:[X]]
[INPUT DATA IN HEX FORM, >FF TO STOP INPUT
[DATA:[5F]]
[DATA:[100]]

[MORE (Y OR N):[Y]]

[EDIT OR FILE:[FILE]]
[FILENAME[PROM1.DAT]]
[FROM ADDRESS IN HEX:[0]]
[OPTION:[VERIFY]]
[INSERT FROM IN SOCKET:[X]]

[EPROM SUCCESSFULLY PROCESSED
[TAKE PROM OUT OF SOCKET:[X]]

[MORE (Y OR N):[N]]

[READY]

77/12/06. 10.17.00.
PROGRAM PROMSRC

```

10 DIM B$(20)
11 FL=0
200 BD=PEEK(-24986)+256*PEEK(-24985):REM-SAVE POINTER TO FREE RAM
301 BD=BD+10
205 CD=BD
207 IF BD>32767 THEN BD=BD-65536
210 AD=61439
220 POKE(-24986),AD-INT(AD/256)*256:REM-SET POINTER TO F000H
230 POKE(-24985),INT(AD/256)
240 LOADPRINT "EPROM.DAT":REM-LOAD I/O DRIVER INTO RAM
250 POKE(-24575),AD+1-INT((AD+1)/256)*256:REM-SET UP START ADDR
260 POKE(-24574),INT((AD+1)/256)
265 POKE(-24986),CD-10-INT((CD-10)/256)*256:REM-RESTORE POINTER
266 POKE(-24985),INT((CD-10)/256)
270 PLOT 12:PRINT "INTEL 2716 PROM PROGRAMMER"
271 GOSUB 900:REM-PRINT INSTRUCTIONS
272 PRINT:PRINT:INPUT"EDIT, FILE, OR READ:";A$
280 IF A$="EDIT" THEN 700
285 IF A$="READ" THEN 5000
290 IF A$<>"FILE" THEN 270
300 INPUT "FILENAME:";A$:REM-***BEGINNING OF FILE PROCESSING***
301 POKE(-24986),CD-INT(CD/256)*256:POKE(-24985),INT(CD/256)
310 LOADPRINT A$:REM-LOAD IN DATA FILE
311 POKE(-24986),CD-10-INT((CD-10)/256)*256:REM-SET UP DCB FOR
312 POKE(-24985),INT((CD-10)/256) :REM-I/O DRIVER
320 POKE BD-9,1
330 POKE BD-9,(CD+3)-INT((CD+3)/256)*256
340 POKE BD-7,INT((CD+3)/256)
350 POKE BD-5,PEEK(BD+2)
360 POKE BD-6,PEEK(BD+1)
370 PRINT "EPROM ADDRESS IN HEX:";GOSUB 2000:B=D4
380 POKE BD-3,B-INT(B/256)*256
390 POKE BD-1,INT(B/256)
400 POKE(-24603),CD-10-INT((CD-10)/256)*256:REM-PASS DCB ADDRESS
410 POKE(-24602),INT((CD-10)/256) :REM-AT LOC 9F00H
420 G=CALL(CD-10):REM-INITIALIZE I/O PORT
430 INPUT "OPTION:";C$
440 IF C$="PROGRAM" THEN 445
450 IF C$<>"VERIFY" THEN 410
460 POKE BD-9,2:REM-VERIFY ONLY
470 GO TO 450
480 POKE BD-9,0:REM-PROGRAM AND VERIFY
490 INPUT "INSERT PROM IN SOCKET:";D$
500 IF C$="PROGRAM" THEN INPUT "TURN ON +25V:";B$
510 G=CALL(CD-10):REM-TRANSFER CONTROL TO I/O DRIVER
520 IF PEEK(BD-10)<>0 THEN 570:REM-I/O SUCCESSFUL?
530 PRINT:PRINT "EPROM SUCCESSFULLY PROCESSED"
540 IF C$="PROGRAM" THEN INPUT "TURN OFF +25V:";B$
550 INPUT "TAKE EPROM OUT OF SOCKET:";B$
560 PRINT:INPUT "MORE (Y OR N)?";C$
570 PRINT:PRINT
580 IF C$="N" THEN 9999
590 IF C$="Y" THEN 530
600 GO TO 570
510 G=PEEK(BD-4)+256*PEEK(BD-3)

```

```

1779 REM-*****ROUTINE TO INPUT HEX CHARACTERS*****
1800 INPUT":":B$
1810 X=LEN(D$):FOR I=1 TO X:D$(I)=MID$(D$,I,1)
1820 IF D$(I)="A" THEN D$(I)="10"
1830 IF D$(I)="B" THEN D$(I)="11"
1840 IF D$(I)="C" THEN D$(I)="12"
1850 IF D$(I)="D" THEN D$(I)="13"
1860 IF D$(I)="E" THEN D$(I)="14"
1870 IF D$(I)="F" THEN D$(I)="15"
1880 NEXT
1890 B4=0
1900 FOR I=1 TO X
1910 D4=D4*16+VAL(D$(I))
1920 NEXT
1930 RETURN
1949 REM *****ROUTINE TO OUTPUT HEX CHARACTERS*****
1950 FOR I=0 TO 15:READ D$(I)
1960 DATA "0","1","2"
1970 DATA "3","4","5","6","7","8","9","A","B","C","D","E","F"
1980 NEXT I
1990 RESTORE
2000 Y2=X/256:X3=256*(X2-INT(X2))
2010 X4=X3016:X5=16*(X4-INT(X4))
2020 X2=INT(X2):X4=INT(X4):X5=INT(X5)
2030 IF FL=1 THEN 4100
2040 PRINT "":D$(X4);D$(X5);
2050 RETURN
2060 PRINT "":D$(X2);D$(X4);D$(X5);
2070 FL=0
2080 RETURN
2099 REM-*****ROUTINE TO READ FROM*****
2100 PRINT"FROM ADDRESS":GOSUB2000:A=D4
2110 INPUT"INSERT FROM IN SOCKET:":B$
2120 POKE BD-2:POKE BD-8,CD+2-INT((CD+2)/256)*256
2130 POKE BD-7,INT((CD+2)/256)
2140 POKE BD-6:POKE BD-5,0
2150 POKE BD+2,05
2160 X=A:FL=0:GOSUB4000:PRINT,;
2170 FOR Z=1 TO 16
2180 POKE BD-2:A=INT(A/256)*256:POKE BD-1,INT(A/256)
2190 POKE (-24608)+CD-10-INT((CD-10)/256)*256
2200 POKE (-24607)+INT((CD-10)/256)
2210 Z=CALL(CD-10)
2220 IF PEEK(BD-10)=0 THEN 6000
2230 IF Z=0 THEN Z=Z+65536
2240 PRINT " ":X=INT(Z/256):GOSUB4000
2250 A=A+1:IF A=2047 THEN 530
2260 NEXT
2270 PRINT
2280 GO TO 5036
2290 G=255:PRINT,;:GOSUB4000
2300 GOTO5100
2310 END
2320 READ.

```



```

500 GOSUB 600
510 GO TO 530
520 REM *****I/O ERROR ROUTINE*****
530 PRINT:PRINT:PRINT:IF PEEK(BD-10)=2 THEN PRINT "VERIFY";
540 IF PEEK(BD-10)=1 THEN PRINT "ADDRESS";
550 PRINT " ERROR"
560 IF PEEK(BD-10)=1 THEN PRINT "ADDRESS";:FL=1:X=Z:GOSUB 4000
570 IF PEEK(BD-10)=2 THEN 630
580 PRINT "FROM ADDRESS AT WHICH ERROR OCCURRED=";
590 Z=PEEK(BD-6)+PEEK(BD-5)*256-CC:FL=1:GOSUB 4000
600 PRINT:PRINT
610 IF PEEK(BD-9)=0 THEN INPUT "TURN OFF +25V:";B$
620 RETURN
630 PRINT "DATA";:X=Z-INT(Z/256)*256:GOSUB 4000:PRINT
640 PRINT "FROM";:X=INT(Z/256):GOSUB 4000:PRINT
650 GOTO 610
660 REM *****EDIT PROCESSING*****
670 PRINT "FROM ADDRESS";:GOSUB 2000:A=D4
680 POKE BD-9,C:POKE BD-8,(CD+2)-INT((CD+2)/256)*256
690 POKE BD-7,INT((CD+2)/256):POKE BD-6,1:POKE BD-5,0
700 INPUT "INSERT PROM IN SOCKET:";B$
710 INPUT "TURN ON +25V:";B$
720 PRINT "INPUT DATA IN HEX FORM, >FF TO STOP INPUT"
730 PRINT "DATA";:GOSUB 2000:C=D4
740 POKE BD-2,A-INT(A/256)*256
750 POKE BD-1,INT(A/256)
760 IF C=0 THEN 270
770 POKE BD-2,C
780 POKE (24608)+CD-10-INT((CD-10)/256)*256
790 POKE (24607)+INT((CD-10)/256)
800 G-CALL(CD-10)
810 IF PEEK(BD-10)<>0 THEN 830
820 A=A+1
830 GO TO 720
840 C=PEEK(BD-4)+256*PEEK(BD-3)
850 GOSUB 600
860 GO TO 530
870 PRINT:PRINT:PRINT:PRINT,,, "INSTRUCTIONS"
880 PRINT:PRINT "1. PLUG PROGRAMMER BOARD INTO THE 24-BIT I/O PORT IN"
890 PRINT " BACK OF THE INTECOLOR 8051"
900 PRINT "2. PLUG THE BOARD INTO AN AC OUTLET. THE POWER ON INDICATOR"
910 PRINT " SHOULD COME ON"
920 PRINT "3. DO NOT TURN ON THE +25V POWER UNTIL INSTRUCTED TO DO SO"
930 PRINT "4. WHEN THE PROGRAM PROMPTS WITH EDIT, FILE, OR READ:, TYPE EDIT IF"
940 PRINT " SINGLE BYTES ARE TO BE MODIFIED, FILE IF AN ENTIRE FILE"
950 PRINT " IS TO BE PROGRAMMED, OR READ IF THE PROM IS TO BE READ AND"
960 PRINT " DISPLAYED."
970 PRINT "5. THE FILENAME MUST BE APPENDED WITH A .DAT EXTENSION"
980 PRINT "6. IF A FILE IS TO BE PROCESSED, THE PROGRAM RESPONDS WITH"
990 PRINT " OPTION AFTER THE FROM ADDRESS HAS BEEN INPUT. THE OPTIONS"
1000 PRINT " ARE PROGRAM (PROGRAM AND VERIFY) AND VERIFY (VERIFY ONLY)"
1010 PRINT "7. WHEN PROMPTED WITH COMMANDS LIKE INSERT PROM IN SOCKET:"
1020 PRINT " RESPOND BY TYPING ANY CHARACTER FOLLOWED BY A CARRIAGE RETURN."
1030 PRINT " JUST TYPING A CARRIAGE RETURN WILL CAUSE A PROGRAM EXIT. TO"
1040 PRINT " RETURN TO THE PROGRAM AT THE POINT OF EXIT, TYPE CONT."
1050 PRINT

```

**EPROM PROGRAMMER DRIVER
DEVICE CONTROL BLOCK**

Attachment 3

STATUS	DB	00H	;	Returned by driver (8 bits)
OPCODE	DB	00H	;	(8 bits)
BUFADR	DW	0000H	;	Address of data (16 bits)
WDCT	DW	0000H	;	No. of words to be processed (16 bits)
CTCT	DW	0000H	;	No. of words not processed (16 bits)
PROMAD	DW	0000H	;	PROM address (16 bits)

STATUS:

00 = Successful completion
01 = Address error (PROMAD \geq 2048)
02 = Verify error (program does not verify)

CTCT:

If status = 00, CTCT = 0. If Status not equal 00, CTCT contains number of words remaining to be processed.

OPCODE:

00 = program and verify
01 = initialize I/O port
02 = verify only

To call driver from BASIC

1. Set LOC A000H to JMP to driver
A000 C300F0
2. Set LOC 9FE0H with address of DCB
(low byte) (high byte)
3. Call driver with CALL(X) statement (Example: Z = CALL(X))

If error condition results:

ADDR error - Z (above) = address at which error occurred
VER error - Z = PROM data *256 + buffer data

```

                                TITLE 'INTEL 2716 EPROM PROGRAMMER DRIVER'
                                ORG 0F000H
F000
;
;***INTEL 2716 EPROM PROGRAMMER DRIVER VERSION-1
;
;***TO BE USED ON INTECOLOR 8051 TERMINAL WITH OPTION 53,
;*** 24-BIT BI-DIRECTIONAL I/O PORT
;
0080      INTC      EQU 80H          ;MODE 0, A,B,C=OUTPUT
0090      PORTA     EQU 90H          ;PORT A ADDRESS
0091      PORTB     EQU 91H          ;PORT B ADDRESS
0092      PORTC     EQU 92H          ;PORT C ADDRESS
0093      CMDWB     EQU 93H          ;COMMAND REGISTER ADDRESS
0020      CINIT     EQU 20H          ;CHIP SELECT HIGH
1100      TMR      EQU 1100H        ;TIMER TO GET 50 NS PULSE
0009      SETC      EQU 00001001B   ;SET PROGRAM STROBE
0008      CLRC      EQU 00001000B   ;CLEAR PROGRAM STROBE
0090      VER       EQU 10010000B   ;VERIFY MODE: A=INPUT
0092      TERM      EQU 10010010B   ;SET I/O PORT TO HIGH Z
000A      CSACT     EQU 00001010B   ;CHIP SELECT ACTIVE
0008      CSINA     EQU 00001011B   ;CHIP SELECT INACTIVE
0001      ERR1      EQU 01H          ;ERR CODE 1-ILLEGAL ADDR
0002      ERR2      EQU 02H          ;ERR CODE 2-VERIFY ERROR
0002      VONLY     EQU 02H          ;VERIFY ONLY OPCODE
0001      IONLY     EQU 01H          ;INIT ONLY OPCODE
2C53      QUIT      EQU 2C53H        ;ROUTINE THAT GOES BACK TO BASIC
;
;DRIVER INITIALIZATION
;
;
F000      ES        PUSH H          ;SAVE ALL REGISTERS
F001      DS        PUSH D          ;          IN STACK
F002      2AE09F     LHLD 9FE0H      ;LOAD IN DCB ADDR
F005      ES        PUSH H          ;SAVE DCB ADDR
F006      23        INX H           ;GET OPCODE
F007      7E        MOV A,H         ;
F00C      FE01      CPI IONLY       ;DECODE OPCODE
F00A      CACBF0     JZ I           ;INIT ONLY
F00B      F2BEF0     JP V           ;VERIFY ONLY
F010      E1        POP H           ;
F011      ES        PUSH H         ;SET UP REGISTERS
F012      CD36F0     CALL GETIF      ;
F015      CD51F0     CALL PROG       ;PROGRAM PROM
F018      E1        POP H           ;
F019      ES        PUSH H         ;
F01A      CD8CF0     CALL VERIFY     ;VERIFY PROGRAM
F01D      AF        XRA A           ;
F01E      F5        PUSH PSW        ;SAVE STATUS
;
;I/O TERMINATION ROUTINE
;

```

```

F01F 3E92  IOTERM: MVI A,TERM      ;SET PORTS TO HIGH Z
F021 D393          OUT CHDMD      ;OUTPUT COMMAND
F023 F1          POP PSW          ;RETRIEVE STATUS
F024 E1          POP H            ;RETRIEVE BCD ADDR
F025 77          MOV M,A          ;STORE STATUS
F026 23          INX H
F027 23          INX H
F028 23          INX H
F029 23          INX H
F02A 23          INX H
F02B 23          INX H
F02C 71          MOV M,C          ;STORE WORD CTR IN CTCT
F02D 23          INX H
F02E 70          MOV M,D

;
;RESTORE REGISTERS AND RETURN
;

F02F 7A          MOV A,D          ;RETURN ERROR DATA IN A AND D
F030 43          MOV B,E
F031 D1          POP D
F032 E1          POP H
F033 C3532C      JMP QUIT

;
;SET UP REGISTERS
;

F036 23  GETBF: INX H
F037 23          INX H
F038 5E          MOV E,H          ;GET LOW BYTE
F039 23          INX H
F03A 56          MOV D,H          ;GET HIGH BYTE
F03B D5          PUSH D           ;SAVE ON STACK
F03C 23          INX H            ;GET CHR COUNT
F03D 4E          MOV C,H          ;LOW BYTE
F03E 23          INX H
F03F 46          MOV B,H          ;HIGH BYTE
F040 23          INX H            ;GET PROM ADDR
F041 23          INX H
F042 23          INX H
F043 5E          MOV E,H          ;LOW BYTE
F044 23          INX H
F045 56          MOV D,H          ;HIGH BYTE
F046 E1          POP H            ;RETRIEVE BUFFER ADDR
F047 C9          RET              ;RETURN

;
;INITIALIZATION ROUTINE
;

F048 3E80  INIT: MVI A,INITC      ;INPUT INIT COMMAND
F04A D393          OUT CHDMD      ;A,B,C-OUTPUT MODE 0
F04C 3E20          MVI A,CINIT    ;SET CHIP SELECT, ALL
F04E D392          OUT PORTC      ;OTHER BITS = 0
F050 C9          RET

```

```

;
;PRGM PROGRAMMER ROUTINE
;
F051 7B      PROG:  MOV  A,E          ;LOAD IN LOW BYTE OF PRGM ADDR
F052 D391    OUT  PORTB          ;OUTPUT LOW ORDER ADDR BYTE
F054 7A      MOV  A,D          ;OUTPUT HIGH ORDER ADDR BYTE
F055 FE00    CPI  00H          ;ADDR 2K?
F057 F2D0F0  JP   ADDRERR        ;YES,ERROR
F05A F620    ORI  CINIT        ;SET CHIP SELECT
F05C D392    OUT  PORTC
F05E 7E      MOV  A,M          ;OUTPUT DATA BYTE
F05F D390    OUT  PORTA
F061 F3      DI
F062 C5      PUSH B          ;SAVE CHR CT
F063 010011  LXI  B,TIMER      ;INITIALIZE TIMER
F066 3E09    MVI  A,SETC      ;SET PG/PPH LINE
F068 D393    OUT  CMDMD        ;OUTPUT COMMAND
F06A 0B      DECR:  DCX  B          ;DECREMENT CTR
F06B 78      MOV  A,B          ;IF B=0,
F06C A7      ANA  A          ;WE ARE ON LAST 256
F06D C26AF0  JNZ  DECR
F070 0B      DECR2: DCR  C          ;THIS INSTR AFFECTS FLAGS
F071 C270F0  JNZ  DECR2        ;LOOP UNTIL DONE
F074 3E08    MVI  A,CLRC      ;CLEAR PG/PPH LINE
F076 D393    OUT  CMDMD
F078 FB      EI
F079 3EFF    MVI  A,OFFH      ;RESET DATA LINES TO ONE
F07B D390    OUT  PORTA
F07D C1      POP  B          ;RESTORE MD CTR
F07E 23      INX  H          ;INCR BUFFER ADDR
F07F 13      INX  B          ;INCR PRGM ADDR
F080 0B      DCX  B          ;DECR MD CTR
F081 78      MOV  A,B          ;CHECK IF MD CTR = 0
F082 A7      ANA  A          ;TEST A
F083 C251F0  JNZ  PROG        ;NO,LOOP UNTIL DONE
F086 79      MOV  A,C
F087 A7      ANA  A
F088 C251F0  JNZ  PROG        ;LOOP UNTIL DONE
F08B C9      RET

;
;VERIFY ROUTINE
;
F08C C836F0  VERIFY: CALL GETDF    ;SET UP REGISTERS
F08F 3E90    MVI  A,VER        ;A=INPUT, B,C=OUTPUT
F091 D393    OUT  CMDMD
F093 7B      VER2:  MOV  A,E          ;OUTPUT LOW ADDR BYTE
F094 D391    OUT  PORTB
F096 7A      MOV  A,D          ;LOAD IN HIGH ADDR BYTE
F097 FE00    CPI  00H          ;ADDR 2K?
F099 F2D0F0  JP   ADDRERR        ;YES, ERROR
F09C F620    ORI  CINIT        ;CHIP SELECT INACTIVE

```

INTEL 2716 EPROM PROGRAMMER DRIVER

```

FO9E B392      OUT  PORTC
FOA0 3E0A      MVI  A,CSACT      ;CHIP SELECT ACTIVE
FOA2 B393      OUT  CNDWD
FOA4 DB90      IN   PORTA      ;READ PROM
FOA6 BE        CNP  N          ;SAME AS BUFFER?
FOA7 F5        PUSH PSW        ;SAVE DATA IN CASE OF ERROR
FOA8 3E0B      MVI  A,CSINA     ;CHIP SELECT INACTIVE
FOAA B393      OUT  CNDWD
FOAC C2B7F0     JNZ  VERERR      ;NO, ERROR
FOAF F1        POP  PSW        ;RESTORE STACK IF NO ERROR
FOB0 23        INX  H          ;INCR BUFFER ADDR
FOB1 13        INX  D          ;INCR PROM ADDR
FOB2 0B        DCX  B          ;DCR WD CTR
FOB3 78        MOV  A,B        ;DONE?
FOB4 A7        ANA  A
FOB5 C293F0     JNZ  VER2       ;NO, LOOP
FOB8 79        MOV  A,C
FOB9 A7        ANA  A
FOBA C293F0     JNZ  VER2       ;NO, LOOP
FOBD C9        RET

;
;VERIFY ONLY
;
FOBE E1        V:      POP  H
FOBF E5        PUSH H
FOC0 CDBCF0     CALL VERIFY      ;CALL VERIFY SUBR
FOC3 AF        DI  A        ;SET UP COMPLETION STATUS
FOC4 F5        PUSH PSW
FOC5 C31FF0     JMP  IOTERM

;
;INIT ONLY
;
FOCB CD4BF0     I:      CALL INIT      ;CALL INIT RTN
FOCB AF        XRA  A          ;SET UP COMPLETION STATUS
FOCC F5        PUSH PSW
FOCD C323F0     JMP  IOTERM+4

;
;ADDRESS ERROR
;
FOD0 F1        ADDERR: POP  PSW      ;GET RID OF PC FROM CALL
FOD1 3E01      MVI  A,ERR1      ;SET UP ERROR STATUS
FOD3 F5        PUSH PSW
FOD4 C31FF0     JMP  IOTERM      ;TERMINATE

;
;VERIFY ERROR
;
FOD7 D1        VERERR: POP  B      ;RESTORE INPUT DATA
FOD8 F1        POP  PSW        ;GET RID OF PC FROM CALL
FOD9 3E02      MVI  A,ERR2      ;SET UP ERROR STATUS
FODB F5        PUSH PSW
FODC SE        MOV  E,H        ;PLACE BUFFER DATA IN A

```

0000 MACRO ASSEMBLER, VER 2.4
INTEL 2716 EPROM PROGRAMMER DRIVER

ERRORS = 0 PAGE 5

F000 C31FF0 JNP I0TERN I0TERMINATE
NO PROGRAM ERRORS END

8080 MACRO ASSEMBLER, VER 2.4
INTEL 2716 EPROM PROGRAMMER DRIVER

ERRORS = 0 PAGE 6

SYMBOL TABLE

* 01

A	0007	ADDER	F0D0	B	0000	C	0001
CINIT	0020	CLRC	0008	CHDND	0093	CSACT	000A
CSIMA	000B	D	0002	DECR	F06A	DECR2	F070
E	0003	ERR1	0001	ERR2	0002	GETDF	F034
H	0004	I	F0C8	INIT	F048	INITC	0080
IONLY	0001	IOTER	F01F	L	0005	M	0006
PORTA	0090	PORTB	0091	PORTC	0092	PROG	F051
PSW	0006	QUIT	2C53	SETC	0009	SP	0006
TERH	0092	TIMER	1100	V	F0DE	VER	0090
VER2	F093	VERER	F0D7	VERIF	F0BC	VONLY	0002 *

AD-A108 253

RAYTHEON CO WAYLAND MA EQUIPMENT DIV

F/G 17/4

R&D EQUIPMENT INFORMATION REPORT. FAULT TOLERANT WEATHER RADAR --ETC(11)

MAR 81 M J YOUNG, A J JAGODNIK

F1962A-7A-C-0113

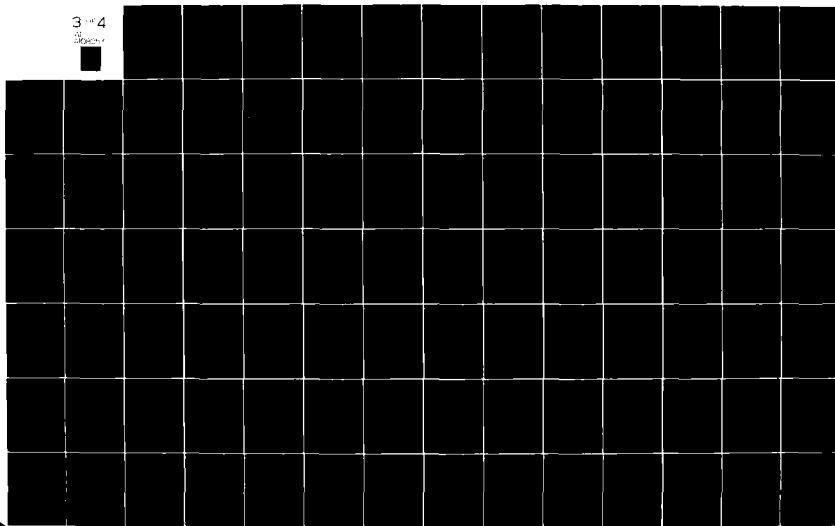
UNCLASSIFIED

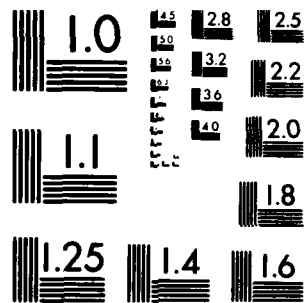
ER81-4093

AFGL-TR-81-0086

NL

3-4





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

A-2

FTWRP Special Instruction Microcode Test Program

FTWRP SPECIAL INSTRUCTION MICROCODE TEST PROGRAM

The microcode test program (FTWRTST) was developed as an aid to debugging the ten special signal processing instructions which were designed for the FTWRP program. It executes as a task in a CE with the task ID of 70 (octal), and reports the results to the Intecolor in the form of a message packet. The program is not meant to be run while the Pulse-Pair system is in operation, but as a diagnostic task to be manually loaded and started using the TSK and ST commands.

FTWRTST performs simple tests of each instruction, and maintains separate error counters for each instruction. Most tests are long enough to ensure that at least one refresh interrupt will occur during execution, thus exercising the interrupt processing microcode. I/O interrupts are not simulated (they are handled differently than refresh interrupts), but they may be invoked by repeatedly sending status request messages to the CE while the task is executing.

The RACOR, RACORI, and RACC tests are performed by sending predefined messages to itself with the proper message codes, and waiting for a flag signifying that the interrupt service routines (RACRIN, RICRIN, and RCIN, respectively) have completed processing the packets. Then the results are compared with expected results to determine if the instructions worked properly. If not, appropriate error counters are incremented and the next instruction test is invoked.

The rest of the instructions are tested by performing the operations on large buffers of predefined data and comparing the results to a similar buffer of expected data. Table A-1 lists the names and functions of each test performed by FTWRTST. The extended SCALE test in Table A-1 (SCLST2) is used to test the SCALE interrupt handling, while the regular SCALE test (SCLSTR) exercises the instruction on a simple case only.

Table A-1. FTWRTST Tests.

<u>Name</u>	<u>Test</u>	<u>Error Counter</u>
START	RACOR	RRERCT
RACSTR	RACC	RCERCT
VADSTR	VADD	VAERCT
VSBSTR	VSUB	VSERCT
CVECST	CVEC	CVERCT
BLNSTR	BLINT	BLERCT
SCLSTR	SCALE	SCERCT
SCLST2	SCALE(extended)	SCERCT
SLNSTR	SLINT	SLERCT
VMSTR	VMULT	VMERCT
RISTR	RACORI	RIERCT
ERCOUT	(outputs error counts to Intecolor)	

A-3

FTWRP Test Target Generator

FTWRP TEST TARGET GENERATOR

The FTWRP test target generator (FTWRTTG) was developed as an aid to algorithm design and test and was used before the Input Synchronizer was fully interfaced to the existing Pulse-Pair Processor. Its main purpose was to generate coherent channel data similar to that produced by actual radar returns so that system data flow and processing could be monitored. It does not attempt to vary the phase shift randomly, and therefore shear calculations are not tested (radial and tangential shear outputs should always be zero).

FTWRTTG executes as a task in a CE under virtual address 51 (octal), and is used in place of the IOC at socket address 13 (octal). Inputs are manually sent to the task via messages from the Intecolor with message code 5. The inputs sent are: reflectivity, and a phase-shift vector in rectangular coordinates, as shown in Table A-2.

Once the parameters have been supplied, a Load Control message (message code 10) is sent to the task with the same format as would be sent to the IOC to enter into the Continuous Input Mode. Thus, the FTWRTTG will appear in the system as a continuous data source similar to the IOC. The task will then send data blocks in the same manner as the IOC, using the virtual address and word count list supplied in the control message. Data will be generated and output forever, until a reset message is received (message code 13).

Table A-3 lists the major subroutines and their functions in the task.

Table A-2. FTWRTTG Parameter Message Format

<u>WORD</u>	<u>CONTENTS</u>
0	Reflectivity value
1	Real component of phase-shift vector
2	Imaginary component of phase-shift vector

Table A-3. FTWRTTG Subroutines

<u>Name</u>	<u>Description</u>
OUTPUT	Sets up header words and outputs block to the current CE
NEXTVA	Determines the next CE to be sent data, and whether coherent or reflectivity data is called for.
LDBUFF	Prepares to generate new coherent channel data buffer if necessary. Generates fake Input Synchronizer "header" words to be put in front of coherent and reflectivity data packets. Calls NEWDAT to generate new coherent channel data.
NEWDAT	Creates new buffer of coherent channel data by performing a complex multiply between the current data buffer elements and the phase-shift vector input by the operator
CMULT	Performs the complex multiply (X register has address of data element, SPEEDI and SPPEDR have imaginary and real components of phase-shift vector)
LCW	Processes the Load Control message from IDOS-1 which supplies the virtual address and wordcount list and starts the generator.
NEWPAR	Processes the parameter message from IDOS-1 which supplies the phase-shift vector and reflectivity data.

A-4

V. E. Follansbee Memo "SEEK IGL00 Common Element Cross Assembler"



FORM 10-0887 (8-68) 8044

DIVISION EQUIPMENT
Operation EDL - WAYLAND
Department SIGNAL PROCESSING

To G. A. Works
From V. E. Follansbee
Subject SEEK IGLOO COMMON ELEMENT
CROSS ASSEMBLER

Classification UNCLASSIFIED

Contract No. _____

Distribution AS LISTED

File No. EM-77-0571

Memo No. VEF:77:14

Date 22 DECEMBER 1977

- References:
1. Specification for Seek Igloo Common Element Cross Assembler RJB-77-98
 2. Selection of Operate and System/Data Processing Instruction Sets for the Common Element RRS:10:77

I. INTRODUCTION

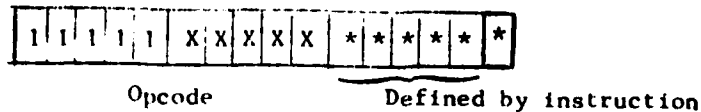
The RP-16 cross-assembler has been modified to provide Common Element assemblies for Seek Igloo. The modified program (MICROL) assembles an instruction set which includes most of the RP-16 code as described in the RP-16 Micro-Minicomputer Programmer's Manual (ER76-4347). This memo describes the changes made to the RP-16 cross assembler and provides a list of possible future improvements, along with the procedure for implementing changes to MICROL on the Cyber system at Bedford. Also included within this memo is a general program description of the Seek Igloo Common Element Cross Assembler.

II. MODIFICATIONS

The following modifications were made to the RP-16 cross assembler. The instruction set for Seek Igloo is shown in Table 1.

1. The Execute instruction (EXEQ) was eliminated from the RP-16 instruction set. If this instruction is encountered, MICROL will output an error message indicating invalid op code and generate a NOP instruction.
2. The Shift Arithmetic Double instruction was given op code value of 23 decimal. This value had been the op code for an EXEQ instruction.
3. The Equivalence instruction (EQU) has the value of its expression field printed in the object code field. There is nothing printed in the location field for an EQU instruction.

4. The Operate group of instructions uses the op code value 31 decimal. The mnemonic forms and values for the second five bit op code field are given in Table 1. The Operate group can eventually expand to 32-64 instructions. The word format for this group is shown below and described in greater detail in reference 2.



5. The register values associated with the P and the S registers has been reversed. Two new register mnemonics have been generated such that registers 6 and 7 are now available to programmers. The following register values are generated by MICROL.

<u>REGISTER MNEMONIC</u>	<u>CODE VALUE (binary)</u>
P	000
S	001
B	010
X	011
A	100
E	101
I	110
W	111

6. The NOP instruction is generated whenever an error condition is encountered. The RP-16 cross assembler had tried to salvage whatever was generated when the error occurred and output that as the object code.
7. The address mode $X+D$ or $BAM=4$ was reassigned to be $BAM=2$. The address mode $((P) + D) + (X)$ or $BAM=2$ was eliminated. This address mode was replaced by a second extended address mode (EAM2). The format of this type instruction is shown below.

1511	10 9 8	7 6	5 4 3	2 1 0
OP Code	1 0 0	EAM2	R1	R2

<u>EAM2</u>	<u>Address Mode</u>	<u>Coding Format</u>
00	Register to Indirect Register	(R2)
01	PreIncremented Pointer, Indirect	(*R1)
10	PreIncremented Pointer	*R1
11	Post Decrement Pointer	R1*

8. The STD, LDD, ADP and SDP instructions will generate an invalid op code error if the W register is used as R2. (i.e. STDW =LABEL)

III. FUTURE IMPROVEMENTS

1. Allow for the option of outputting location and object code information in hexadecimal format. Presently, MICROL outputs this information in an octal format.
2. Output the object code for instructions in a field oriented format. The various fields given in any instruction may be op code, BAM value, EAM value, displacement, R1 or R2.
3. Include a double precision define constant instruction. Double precision words for Seek Igloo have a sign bit in both the most significant and the least significant halves of the word.
4. Develop a macro capability.
5. Add new operate instructions as they are defined (i.e. GETCLK, SETCLK).

IV. IMPLEMENTING MODIFICATIONS

This section describes the procedure for modifying the MICROL program. This procedure utilizes the Cyber system at Bedford and is submitted via card deck at the Information Processing area.

CARD COLUMNS

1...
 NAME. comment area after period
 USER, your user number, your password, SYSTEM2.
 CHARGE, DSO to be charged.
 GET(MICROL)
 MODIFY (P=MICROL,N,C,F,I,O = EM)
 REPLACE (NPL = MICROL)
 REWIND (COMPILE)
 FTN (I = COMPILE, L = 0) remove , L=0 if want a listing produced
 REPLACE (MICRO)
 LOAD 'LGO' }
 NOGO. } ① if want to execute program
 MICRO (INPUT,OUTPUT) in same run as modifications are done.
 6/7/8 EOR card
 *DECK deck name
 *IDENT date
 *I card identification Modify directives
 card to be inserted
 *D card identification
 6/7/8 } ② if requested execution
 code to be assembled EOR card
 6/7/8 EOI card
 \$\$

V. PROGRAM DESCRIPTION

This section provides a brief description of the various programs, subroutines, and functions defined in the MICROL program. MICROL is an overlay program on the CYBER system and has seven overlay segments. The main overlay segment, MICROL, calls the other six overlay segments. These six segments call the various subroutines and functions. A block diagram of MICROL is provided in Figures 1 and 2.

1. Overlay Segments

MICROL	This segment calls the remaining overlay segments in the order listed below. It also outputs the symbol table to TAPE14 if the user has so specified. Some of the remaining overlay segments are executed on a conditional basis. The conditional information is set in the INFORM segment.
INFORM	This segment reads one card with a two digit octal number in card columns 1 and 2. This octal number allows for six program options. These options are described in Table 2.
PASONE	This segment analyzes card images and reduced card images and builds a symbol table based on the label field and a location counter.
HEADER	This segment outputs a header page if the list option has been set by the INFORM segment. Presently, the header is simply a page feed.
PASTWO	This segment analyzes reduced card images and builds the object code. The object code is determined by the operation mnemonic field (OPN) and the argument field on each reduced card image. The symbol table is used if a label is present in the argument field. It should be noted that the argument field is also referred to as the expression field.
SEMOUT	This segment will output a symbol table if the option has been set by the INFORM segment.
SUOUT	This segment will output a set/used reference table if the option requesting a symbol table has not been set.

2. Subroutines

INPUTA	This subroutine is called by PASONE to read and analyze one card image. If it is a /R card, then a flag is set to indicate that reduced card images follow. Otherwise, the label field is analyzed and the symbol table updated accordingly. Also, the operation mnemonic field is analyzed and the location counter updated. A reduced card image is produced for each card and saved in the array SECTOR for use by PASTWO.
INPUTB	This subroutine reads a buffer of reduced card images and moves one card image into the working area TOKEN. Each successive call to this routine will move the next card image from the buffer to the working area. The subroutine OP is called for each card image before the return from INPUTB.
DFINER	This subroutine performs initialization before any symbol table update. If a label exists, then the subroutine DEFINE is called to update the symbol table.
DEFINE	This subroutine saves the label and the associated location counter in the symbol table. If the label was associated with an EQU instruction, then the value of the expression field is saved with the label.
REFER	This subroutine analyzes the expression field of an instruction. The set/used references in the symbol table will be updated and the location counter updated based on the analysis.
SETUSE	This subroutine is called by REFER to add a set/used reference to the symbol table for a given label.
EVALU8	This subroutine initiates evaluation of the expression field. The actual evaluation is performed by the XPRESS subroutine.
XPRESS	This subroutine evaluates the expression field and returns a parameter which reflects the value of the expression field.
NUMRAL	This subroutine decodes any number sequence that may be defined in the expression field. The value of the number is returned to the calling routine. The types of numbers allowed are BAM, octal, and decimal.

Subroutines (Continued)

UNPACK	This subroutine moves a three character sequence from one word into three separate variables.
SEARCH	This subroutine searches the symbol table for a label. The label is specified by pointers passed in the subroutines parameter list. The results of the search are returned to the calling routine in the variable FOUND.
OP	This subroutine evaluates the operation mnemonic field and returns an index for the operation. This index will be used in PASTWO to retrieve the correct object code from the array MCODE.
ROTSOR	This subroutine packs three characters from a card image into one word. This word is then saved as part of the reduced card image format.
BLNKCT	This subroutine counts the number of blanks between the various fields on a card image. This count is saved as part of the reduced card image format.
LITRAL	This subroutine is called by the INPUTA subroutine when processing a TEXT operation.
OBJECT	This subroutine will output the object code to TAPE5 if the user has so specified through the INFORM subroutine.
OUTPUT	This subroutine will output the location value, object code and associated card image if the user has so specified through the INFORM subroutine.
ERROR	This subroutine is called by a number of the other subroutines whenever an error is detected. The number of errors in this assembly and information related to the present error is saved in this routine. If the user has requested output, then ERPRNT is called to output the error message.
ERPRNT	This subroutine outputs an error message to the line printer based on the parameter passed to the subroutine.

3. Functions

HSCAN (CHAR, INDEX, FIRST, LAST)

This function will return a value of TRUE if the character specified by CHAR is found within a string of characters. The string of characters available for the compare are defined in the common area /HSPEC/. The actual characters used for comparing are defined by the parameters FIRST and LAST. The position of CHAR within the string is returned in the variable INDEX.

ISHFT (VAR, NUM)

This function performs a left circular shift on the sixty bit variable VAR. The shift is for NUM bit positions.

LIMIT (ICHAR)

This function returns a value of TRUE if the character specified by ICHAR is a number or an apostrophe.

CHECK (LARG)

This function returns a value of TRUE if LARG has a value between +127 and -128.

NSCAN (IDUM)

This function returns a value of TRUE if any invalid characters are found within the label field. The characters listed below are considered invalid.

\$ # [] : - ' < > ;

TABLE 1

<u>MNEMONIC</u>	<u>OPCODE</u>	
	<u>OCTAL</u>	<u>DECIMAL</u>
1. STS	00	0
2. STD	01	1
3. LDS	02	2
4. LDD	03	3
5. LDC	04	4
6. LDN	05	5
7. SWP	06	6
8. ADD	07	7
9. ADP	10	8
10. SUB	11	9
11. SDP	12	10
12. AND	13	11
13. IOR	14	12
14. XOR	15	13
15. MPY	16	14
16. DIV	17	15
17. JPZ	21	17
18. JNG	22	18
19. JEX	23	19
20. JNZ	24	20
21. CSL	32	26
22. CSE	33	27
23. ASZ	34	28
24. OSF	35	29
25. RSP	36	30
26. SAS	27	23
27. SAD	27	23
28. SET	14	12
29. CLR	13	11
30. SBZ	34	28
31. SBS	35	29
32. NOP	00	0
33. ***	00	0

<u>MNEMONIC</u>	<u>OPCODE</u>	
	<u>OCTAL</u>	<u>DECIMAL</u>
34. ORG	00	0
35. EQU	00	0
36. END	00	0
37. JUMP	20	16
38. JOVF	25	21
39. JSUB	26	22
40. EXEQ	00	0 NOP
41. ISEZ	30	24
42. DSEZ	31	25
43. EHII	27	23
44. ELOI	27	23
45. DLOI	27	23
46. SOVF	27	23
47. ROVF	27	23
48. REXT	27	23
49. HALT	27	23
50. CALL	26	22
51. EXIT	20	16
52. TEXT	00	0
53. END\$	00	0
54. EJCT	00	0
55. IDEN	00	0
56. HOLD	00	0
57. BLOK	00	0
58. BASE	00	0
59. DATA	00	0
60. DS	00	0
61. DC	00	0
62. STB	10	8
63. LDB	30	24
64. ADB	10	8
65. SBB	12	10
66. CEB	17	15

Table 1 Continued

<u>INSTRUCTION SET</u>	
<u>OPCODE</u> <u>DECIMAL</u>	<u>MNEMONIC</u>
0	SELINS
1	SETVAD
2	TRAP
3	DROP
4	TRACE
5	RETURN
6	SYSREQ
7	AVAILABLE
8	INTERR
9	WRITE
10	READ
11	RESUME
12	SELBUS
13	READR
14	WRITER
15	REQIOS
16	CLOCK
17	LOGTST
18	MEMIST
19	SENDIN
20	STARTU
21	TRPRET
22	AVAILABLE
23	AVAILABLE
24	RESTRA
25	
26	
27	

Set up 9 binary digits

MSB

LSB

1 2 3 4 5 6 7 8 9

BIT	EFFECT IF SET	EFFECT IF NOT SET
1	Symbol table output to the file TAPE14 at the end of assembly	Symbol table cannot be used again
2	Outside input for symbol table is used from TAPE15	Symbol table generated by program
3	Errors listed during first pass	No errors listed during first pass
4	Plain symbol table	Set/Used reference table
5	No object code file	Object code file generated and saved to TAPE5
6	No listing	Listing
7	One fill on BLOK statements	Zero fill on BLIK statement
8	Storage blocks allocated by DS instructions will be 1s filled	Storage allocated will be 0s filled
9	Next line of input will have decimal number specifying number of lines per page of listing output	Number of lines per listing page defaults to 51

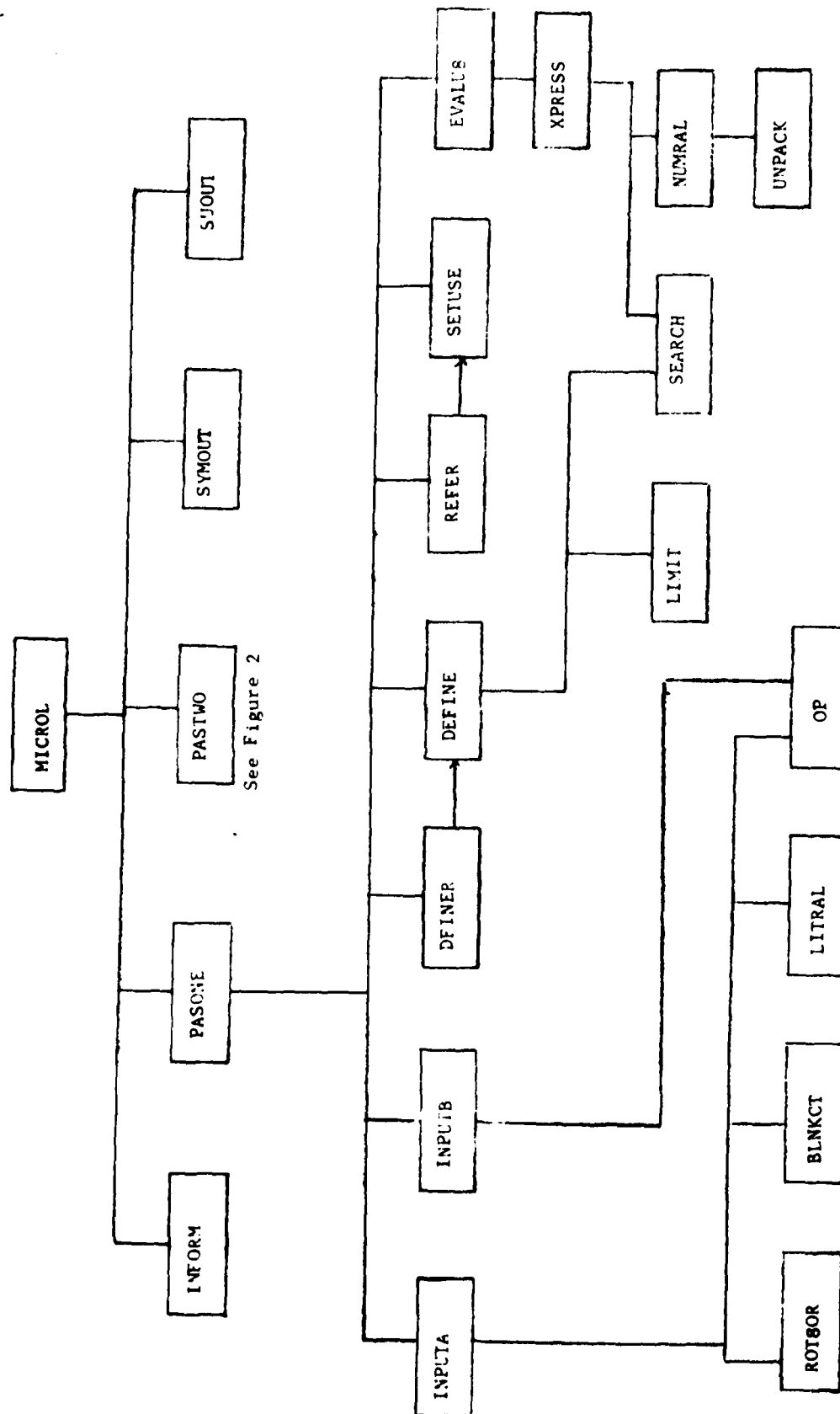


FIGURE 1

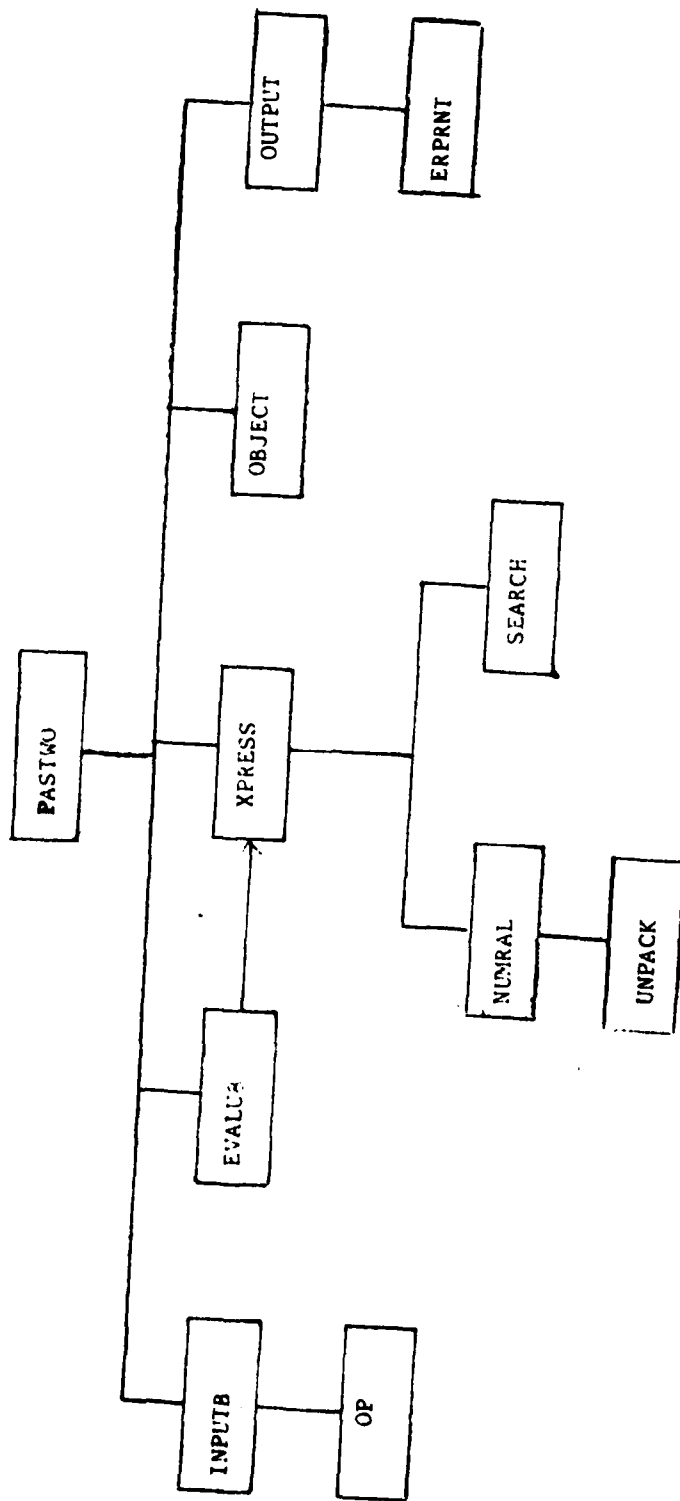


FIGURE 2

UNCLASSIFIED
EM- 77-0571
VEF:77:14
22 DECEMBER 1977
PAGE 10

V. E. Follansbee

V. E. Follansbee
SIGNAL PROCESSING DEPARTMENT
Extension 5340, Wayland
Box F-27

/sjg

Distribution

P. C. Barr
R. J. Bonneau
R. H. Daly
J. M. Glass
J. P. Hepp
M. A. Jones
R. A. Kudlich
D. C. Schleher
R. R. Smith
M. J. Young
Document Control. (2)

A-5

V. E. Follansbee Memo, "Common Element Cross Assembler Post Processor"



FORM 100-1557 (3-65) 4-70

DIVISION EQUIPMENT
Operation EDL - WAYLAND
Department SIGNAL PROCESSING

To G. A. Works

From V. E. Follansbee

Subject COMMON ELEMENT CROSS ASSEMBLER
POST PROCESSOR

Classification UNCLASSIFIED

Contract No. _____

Distribution AS LISTED

File No. EM#78-0004

Memo No. VEF:78:01

Date 06 JANUARY 1978

- References: 1. Seek Igloo Common Element Cross Assembler VEF-77-14
2. Common Element Cross Assembler Post Processor
Specification, RJB:77:102

A post processor program (CEPOSTP) is available for Seek Igloo assemblies. This program converts the binary output from the Cross Assembler program (MICRO) to a format required by the Intecolor terminal transfer programs presently being developed. The binary output from the Cross Assembler is written on TAPE5 and must be saved to a permanent file at the completion of a MICRO execution. This permanent file is then assigned to TAPE5 as an input to the CEPOSTP program. An example of the job stream required when the MICRO and CEPOSTP programs are executed separately is given in Figure 1. The two programs may be executed in the same job stream and this sequence is shown in Figure 2.

There are two types of assembled programs that the CEPOSTP program may process. These are DOSO type programs and CE software programs. For DOSO programs, two output tapes are generated by CEPOSTP. The first tape (TAPE8) contains one record representing the 8 MSBs of all the assembled code. The second tape contains the 8 LSBs of all the code and is saved to TAPE9. These tapes should be saved to permanent files for access by the Intecolor terminal transfer program. For CE software programs, all output is written to TAPE8 and should be saved to a permanent file at the completion of CEPOSTP.

The CEPOSTP program requires three inputs from the user. These inputs are the ID number of the program being processed, the type of program, and a comment field. The ID must be specified in the first three columns of the first input card. The type of program is specified in column five of the first card. The types presently defined are 0 = CE program, and 1 = DOSO program. Column seven of the first card is an optional parameter. If a one is set in this column, the character sequence written to TAPE8 (and TAPE9 if DOSO program) is dumped to the line printer. The comment field is read from the first 70 columns of a second input card. This comment field is output to the line printer before any error or processing messages.

UNCLASSIFIED
EM#78-0004
VEF:78:01
06 JANUARY 1978
PAGE 2

JOB CARD
USER CARD
CHARGE CARD
GET (MICRO/UN=R589201)
MICRO (INPUT, OUTPUT, TAPE5)
REWIND (TAPE5)
SAVE (TAPE5=NAME 1)
EOR CARD
OPTION CARD
CODE TO BE ASSEMBLED
EOI

JOB CARD
USER CARD
CHARGE CARD
GET (CEPOSTP)
GET (TAPE5=NAME 1)
CEPOSTP (INPUT, OUTPUT, TAPE5, TAPE8, TAPE9)
REWIND (TAPE8)
SAVE (TAPE8=NAME2)
REWIND (TAPE9)
SAVE (TAPE9=NAME3) } Only need these 2 if DOSO code
EOR CARD
ID/TYPE/DUMP OPTION (INPUT CARD)
COMMENT INPUT CARD
EOI CARD

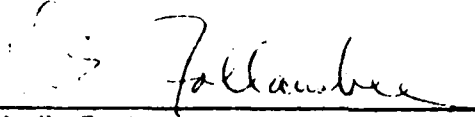
FIGURE 1

UNCLASSIFIED
EM#78-0004
VEF:78:01
06 JANUARY 1978
PAGE 3

JOB CARD
USER CARD
CHARGE CARD
GET (MICRO/UN=R589201)
MICRO (INPUT, OUTPUT, TAPE5)
GET (CEPOSTP/UN=R589201)
CEPOSTP (INPUT, OUTPUT, TAPE5, TAPE8, TAPE9)
REWIND (TAPE8)
SAVE (TAPE8=NAME1)
REWIND (TAPE9) } — only need these 2 if DOSO code
SAVE (TAPE9=NAME2)
EOR CARD
OPTION CARD
CODE TO BE ASSEMBLED
EOR CARD
ID/TYPE/DUMP OPTION/INPUT CARD
LABEL OR COMMENT INPUT CARD
EOI CARD

FIGURE 2

UNCLASSIFIED
EM#78-0004
VEF:78:01
06 JANUARY 1978
PAGE 4


V. E. Follansbee
SIGNAL PROCESSING DEPARTMENT
Extension 5340, Wayland
Box F-27

/sjg

Distribution

See Attached Sheet

A-6

V. A. Jelich Memo, "Cyber/Intecolor Support Software"

COMPANY PRIVATE

A-7

V. E. Follansbee Memo

"Intecolor Terminal Transfer Program Description"



FORM 10-0957 (9-65) 0040

Classification UNCLASSIFIED

DIVISION EQUIPMENT
Operation EDL - WAYLAND
Department SIGNAL PROCESSING

Contract No. _____

Distribution AS LISTED

To G. A. Works

File No. EM#78-0013

From V. E. Follansbee

Memo No. VEF:78:03

Subject INTECOLOR TERMINAL TRANSFER
PROGRAM DESCRIPTION

Date 10 JANUARY 1978

Reference: FTSP Common Memory Organization, RJB-77-100

A program is required to transfer a file from the CYBER system to the Intecolor terminal. This transfer program must handle three types of input files from the CYBER. These types are:

1. 8-bit PROM programs
2. 16-bit DOSO PROM programs
3. 16-bit CE software programs

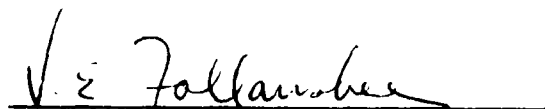
The formats for these files are shown in Figures 1, 2 and 3 respectively.

A program presently exists which transfers the first and second types of files to the terminal. When initiated, this program requests a CYBER file name from the user. The program will transfer each record on the CYBER file into the terminal memory at the address specified in the record. The transfer program recognizes the last record by a zero byte count. Once this has been detected, a request is made for a disk file name. The data in the terminal memory is then saved to that file name. The save to disk is executed via a File Control System command. The transfer program is written in BASIC and uses various 8080 programs along with the FCS command.

When CE software programs are transferred from the CYBER, more than one disk file needs to be saved. Instead of transferring all the records into memory and then performing a save to disk, the save must be executed after each record. The data saved after each record is the 240 16-bit words. Also, two disk files must be generated which contain information about the program transferred and what disk files were used to save the program. This disk structure differs enough to require a separate transfer program for CE software programs. The disk structure used will conform with the Common Memory page format description in the above referenced memo. To accomplish this, all disk files will be given names of PAGE.### where ### is greater than ten and determined by the transfer program. The disk file PAGE.003 will be the directory page and its format is shown in Figure 4. The directory page provides a list of the various CE software programs that may have been transferred. The status and ID parameters in each

EM#78-0013
VEF:78:03
10 JANUARY 1978
PAGE 2

entry are defined at transfer time (see Figure 3, 12-15). The Load Map Page parameter is the number in the PAGE.### of a disk file name. This disk file contains information about the program transferred and a list of disk files that the program was saved to. The information contained on a Load Map Page disk file and its format is shown in Figure 5. The header words on the Load Map Page are set by the transfer program. The last three parameters in each segment of this disk file are defined at transfer time. The Page No. (###) parameter is computed by the transfer program. To determine which page numbers have been used, a disk file must be maintained with a list of free and used numbers. All saves of the various types of disk files is controlled by the transfer program. There is no requirement that the user know what files are available. The Load Map Page disk file number will be output to the user when the program has completed.



V. E. Follansbee
SIGNAL PROCESSING DEPARTMENT
Extension 5340, Wayland
Box F-27

/sjg

Distribution

R. J. Bonneau
R. H. Daly
G. A. Sarafinas
M. J. Young
Document Control (2)

ASCII Characters on a record of the CYBER file

1.	:	
2.	}	byte count
3.		
4.		
5.		
6.	}	starting address
7.		
8.		
9.		
10.	0	
11.	0	
12.	}	first byte
13.		
14.		
.		
.		
.		
.		
.		
n-2	}	last byte
n-1		
n	}	checksum*
n+1		

*The checksum is the two's complement of the sum of all the data after the colon and before the ascii conversion.

FIGURE 1

ASCII Characters on CYBER file TAPE8 after CEPOSTP

```

1.      :
2.
3.  }   byte count
4.  }
5.
6.  }   starting address
7.  }
8.  }
9.
10.     0
11.     2
12.  }   8 MSBs of first 16 bit word  *1
13.  }
14.  }   8 MSBs of second 16 bit word
15.  }
16.
17.
.
.
.
n-2. }   8 MSBs of last 16 bit word
n-1. }
n     }   Checksum *2
n+1. }

```

*2 This checksum is the twos complement of the sum of everything after the colon and before the ascii conversion was made.

*1 The 8 LSBs of every 16 bit word are saved on TAPE9 at the end of a CEPOSTP execution. This tape is formatted the same as above.

FIGURE 2

16-Bit DOSO Program Format

ASCII Characters on each record of the CYBER

1.	:
2-5.	byte count = 488
6-9.	starting address
10.	0
11.	1
12-13.	✓ status
14-15.	✓ id
16-19.	✓ starting address
20-23.	✓ word count (= 240)
24-27.	✓ 16-bit checksum *1
28-31.	first 16-bit word
32-35.	
.	
.	
.	
980-983.	239th 16-bit word
984-988.	240th 16-bit word
989-990.	8-bit checksum *1

*1 Twos complement of the sum of all the 16-bit words on this record.

*2 Twos complement of the sum of everything after the colon. The sum is computed before the ascii conversion is implemented.

FIGURE 3

16-Bit CE Software Program Format

Entry

	Word 0	Word 1
1	Status/ID	Load Map Page
2	Status/ID	Load Map Page
3	.	.
	.	.
	.	.
	.	.
128		

FIGURE 4

Directory Page

	Word 0	Word 1	Word 2	Word 3
Header	Map Type	Number of Segments	Identifier	Spare
Segment 1	Page No.	Start Address	Number of Words	Checksum
Segment 2	"	"	"	"
Segment 3				
Segment 4				
.				
.				
.				
			•	
			•	
			•	
Segment 63				

Figure 5. Load Map Format

A-8

R. J. Bonneau Memo

"INODDOS-Utility Program for Inspecting and Modifying DFTSP DOS-0 Object Code"

COMPANY PRIVATE

A-9

R. J. Bonneau Memo, "NEWMOD-MODDOS Enhancement"



FORM 10-0557 (9-68) BOMB

DIVISION EQUIPMENT
Operation EDL - Wayland
Department Advanced Development Laboratory

To A. Bachman

From R. J. Bonneau

Subject NEWMOD - MODDOS Enhancement

Classification Unclassified

Contract No. 79D-320

Distribution EDL-94 "S" List

File No. EM79-0632

Memo No. RJB-195

Date 30 October 1979

Reference 1: EM78-0427, RJB-124, "MODDOS - Utility Program for Inspecting and Modifying DFTSP DOS-0 Object code", 2 August 1978.

I. Introduction

MODDOS (See Reference 1) is a program used to inspect and modify the format for DOS-0 programs used in the Fault Tolerant Signal Processor (FTSP). NEWMOD is an enhancement of that program which is compatible with MODDOS but includes the extra function of a 4K checksum algorithm. This memo describes the checksum algorithm, the usage within NEWMOD and also provides a current listing of the NEWMOD program. NEWMOD shall replace MODDOS on the utility disks.

II. Checksum Algorithm

The Algorithm (See Figure 1) consists of the sequential summing of 4096 16-bit 2's complement values which constitute the 4K address space for the DOS-0 PROM's. In addition, for each sum which results in an overflow condition (e.g. 2 positive numbers added, resulting in a negative number), an overflow counter is incremented. After all 4K words have been summed, the overflow counter is also added in to yield the full checksum value. This value is now 2's complemented (i.e. negated) and stored into the location 177727 (octal) - the checksum location.

The net effect is that the sum of all words, including the checksum location, and the overflow counter, should be exactly 0.

III. Usage

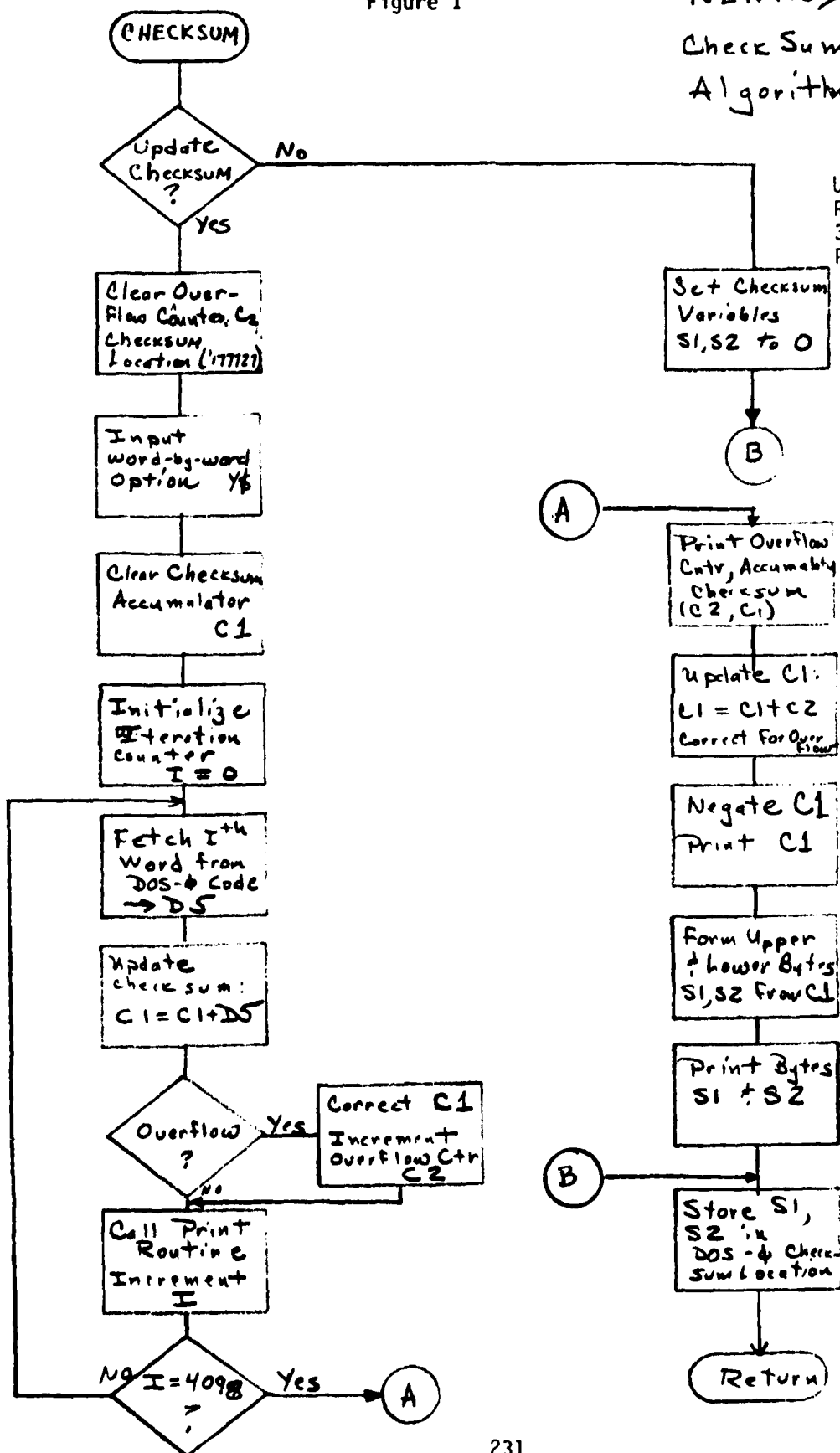
NEWMOD operates exactly as MODDOS as far as DOS-0 memory accessing and modifications are concerned. Once the user has indicated that all changes have

Figure 1

NEWMOD Check Sum Algorithm

10/31/79

Unclassified
RJB-195
30 October 1979
Page 2



been entered and that the updated files shall be saved, NEWMOD now asks:

DO YOU WANT THE CHECKSUM UPDATED?

If the user enters N (for No), NEWMOD will zero the checksum location, then proceed to store the updated files. If the user enters Y, NEWMOD then asks:

DO YOU WANT A WORD-BY-WORD LIST?

Answering Y to this question permits the user to receive a word-by-word listing (on the screen) of each location being summed, its address, the value stored at the location, the current accumulated checksum, and the current overflow counter.

As the checksum is being performed, NEWMOD outputs a message after each 100 words:

XXXX WORDS SUMMED...

After the complete checksum is performed, NEWMOD prints out the following information:

OVERFLOW COUNTER = nnnn (octal)
ACCUMULATED CHECKSUM VALUE = nnnnnn (octal)
NEGATIVE ACCUMULATED CHECKSUM = nnnnnn (octal)
UPPER BYTE = uuu (decimal) LOWER BYTE = lll (decimal)

After this output has been performed, NEWMOD prints out the message:


IF YOU WISH TO CONTINUE, TYPE CONT

and types READY.

The user now enters CONT to continue the processing which results in the storage of the checksum value and the saving of the 4 DOS-0 object files.

IV. Current Listing

See Figure 2 for a complete listing of the current NEWMOD program.


R. J. Bonneau
Advanced Electronic Techniques
Wayland Box M9 x5172

RJB/ema

cc: EDL-94 "S" List

NEWMOD
ON RJB-0
10/11/79
VN9

Figure 2

Unclassified
RJB-195
30 October 1979
Page 4

```
1 PLOT 12:ELLAR 200:DIM B$(16)
2 PRINT""PRINT"      COMMON ELEMENT BOOTSTRAP/DOS-0 NOBILITY PROGRAM"
3 PRINT""
4 FOR I=0 TO 15: READ B$(I):NEXT
5 DATA "0","1","2","3","4","5","6","7","8","9","A","B","C","D","E","F"
6 PRINT""PRINT""PRINT" THIS PROGRAM PERMITS THE USER TO DISPLAY OR MODIFY"
7 PRINT" COMMON ELEMENT BOOTSTRAP OR DOS-0 PROGRAMS STORED ON 1"
8 PRINT" REQUIRES AN MSB (MOST SIGNIFICANT BYTE) AND LSB (LEAST SIGNIFICANT BYTE)"
9 PRINT" FILE"
10 INPUT "ENTER MSB 0 FILENAME:";A$;A$=A$+".DAT"
12 INPUT "ENTER LSB 0 FILENAME:";B$;B$=B$+".DAT"
14 INPUT "ENTER MSB 1 FILENAME:";C$;C$=C$+".DAT"
16 INPUT "ENTER LSB 1 FILENAME:";E$;E$=E$+".DAT"
20 PLOT 27:PRINT"BLQA 1:";A$;" 0000";PLOT 27:PLOT 27
27 PLOT 27:PRINT"BLQA 1:";B$;" 0802";PLOT 27:PLOT 27
28 PLOT 27:PRINT"BLQA 1:";C$;" 0804";PLOT 27:PLOT 27
29 PLOT 27:PRINT"BLQA 1:";E$;" 0806";PLOT 27:PLOT 27
30 INPUT "ENTER STARTING ADDRESS (DECIMAL) >= 170000:";WH
31 ZZ=WH:WH=WH-170000:IF WH<4000 THEN 35
33 WH=WH+2
34 IF WH<4000 THEN 35:WH=WH+2
35 GOSUB 9000:REM CONVERT WH FROM DECIMAL TO BINARY
36 PRINT"      ADDRESS      CONTENTS      NEW VALUE (DECIMAL, BLANK OR -1)"
40 AH=-16384+WH:REM **** 11 BIT MSB BYTE FROM (-32000) + WH
45 AE=-12782+WH:REM **** 11 BIT LSB BYTE FROM (-32004) + WH
48 PRINT TAB(3);ZZ;TAB(20);
50 X=256*PEEK(AH)+PEEK(AH+1):GOSUB 5000
55 PRINT V;
60 PRINT TAB(45);
62 GOSUB 2000
65 IF NV=-1 THEN 80
70 IF NV=2 THEN 100
72 N1=INT(NV/256):POKE AH,N1
74 N2=NV-256*INT(NV/256):POKE AE,N2
80 AH=AH+1:AE=AE+1:REM *****INCREMENTING BUFFER ADDRESS*****
85 WH=WH+1:REM *****INCREMENTING FROM ADDRESS
86 X=WH:IF (WH-2048)*(WH-2049)<0 THEN 88
87 WH=WH+2:X=X+2
88 X=WH:IF WH<2050 THEN 90
89 X=X-2
90 GOSUB 5000
91 ZZ=V+170000:GOTO 40
100 INPUT "ANY MORE CHANGES TO BE MADE ? (Y OR N)";Z$
110 IF Z$ = "Y" THEN 30
115 IF Z$ <> "N" THEN 100
120 INPUT "WRITE OUT THE UPDATED FILLS ? (Y OR N)";Z$
122 IF Z$ = "Y" THEN 125
123 IF Z$ <> "N" THEN 120
124 GOTO 200
125 REM GOSUB 8000:REM CALL THE FROM LIBRARY ROUTINE
130 X=PEEK(-16384)+PEEK(-16383)*256
```

```
140 PLOT127:PRINT"BSAV 1:"IA$1" 0000 "GUSUB4000:PLOT127:PLOT127
150 PLOT127:PRINT"BSAV 1:"IB$1" 0004 "GUSUB4000:PLOT127:PLOT127
160 PLOT127:PRINT"BSAV 1:"IC$1" 0008 "GUSUB4000:PLOT127:PLOT127
170 PLOT127:PRINT"BSAV 1:"IE$1" 0006 "GUSUB4000:PLOT127:PLOT127
180 PRINT" FILES: "A$B$C$E$ "SAVED TO DISK"
200 PRINT"*****END OF MODBUS PROGRAM*****".END
2000 INPUT"ID$"
2003 IF VAL(D$)<>-1 THEN 2005
2004 NV=-2:RETURN
2005 IF LEN(D$)<>0 THEN 2010
2008 NV=-1
2009 RETURN
2010 QQ=WD:WD=VAL(D$):GUSUB 9000:NV=WD:WD=QQ:RETURN
4000 REM *****ROUTINE TO CONVERT BINARY TO ASCII HEX*****
4050 X2=X/256:X3=256*(X2-INT(X2))
4060 X4=X3/16:X5=16*(X4-INT(X4))
4070 X2=INT(X2):X4=INT(X4):X5=INT(X5)
4080 PRINTD$(X2);D$(X4);D$(X5);
4085 PRINT""
4090 RETURN
5000 FOR I=1 TO 6:D(I)=X-8*INT(X/8):X=INT(X/8):NEXT
5010 V=0:FORI=1 TO 6:V=10*V+D(7-I):NEXT
5020 RETURN
8000 REM *****DOS-0 CHECKSUM COMPUTATION AND STORAGE*****
8002 PRINT" FROM CHECKSUM BEING COMPUTED...."
8005 C1=0: C2=0
8010 FOR I=2 TO 2047
8020 C1=C1+PEEK(-16384+I):C2=C2+PEEK(-12286+I)
8030 NEXT
```

Unclassified
RJB-195
30 October 1979
Page 6

```
8040 PRINT "LOWER BYTE SUM=";C1,"LOWER BYTE SUM=";C2
8050 X3=INT(C1/65536);Y=C1-65536*X3;X2=INT(Y/256);Z1=INT(Y/256)
8060 Y3=INT(C2/65536);Y=C2-65536*Y3;Y2=INT(Y/256);Z2=INT(Y/256)
8070 Z1=Y1
8080 Z2=X1+Y2;Z2=INT(Z2/256);Z2=Z2-256*Z2
8090 Z3=X2+Y3;Z3=INT(Z3/256);Z3=Z3-256*Z3
8095 Z4=X3+Z2
8100 PRINT "C1,X3,X2,X1=";C1,X3,X2,X1
8110 PRINT "C2,Y3,Y2,Y1=";C2,Y3,Y2,Y1
8120 PRINT "Z4,Z3,Z2,Z1=";Z4,Z3,Z2,Z1
8130 Z=(Z4*256+Z3)*(Z2*256+Z1)-Z-INT(Z/65536)*65536
8140 PRINT "FINAL CHECKSUM VALUE Z=";Z
8160 RETURN
9000 FOR I=1 TO 6:WHILE=WD:FOR J=1 TO 10:WD=INT(WD/10);NEXT J
9010 WD=0:FOR J=1 TO 6:WD=INT(WD/10);NEXT J
9020 RETURN
```

A-10

M. J. Young Memo, "Modifying Files on the Intecolor"



FORM 10-0587 (9-85) 8040

DIVISION	EQUIPMENT	Classification	Unclassified
Operation	EDL - Wayland	Contract No.	
Department	Advanced Development Laboratory	Distribution	As Listed
To	R. J. Bonneau	File No.	
From	M. J. Young	Memo No.	MJY-03
Subject	Modifying Files on the Intecolor	Date	26 January 1978

To aid the debugging of CE microcode and DOS-0 EPROMs, a program to modify floppy disk files has been written. With the use of this program, data can be modified on the disk without the need for re-assembly and transfer from the Cyber.

The program, called "MODIFY", resides on the CE system disk (CESYSH18) as a BASIC file. To run the program:

1. Enter Basic by typing ESCAPE E
2. Place the system disk in drive 0 (left drive)
3. Place the disk on which the file resides in drive 1
4. Type LOAD?"MODIFY"
5. When the system responds with READY, type RUN.

The program will then ask for the name of the file to be modified. The full name must be specified (Example: CEM1.DAT). Once the file is read into RAM, the program will ask for the word address to be modified. Words are numbered from 0 to (2047)₁₀. Therefore, if the fourth word of a file is to be modified, the word address is 3.


The program will respond by displaying the data in the word specified in hexadecimal notation, and wait for new data. If the word is not to be changed, simply type in the same number. Otherwise, enter the new data in hexadecimal notation. The word will be changed, and the next sequential word displayed as before. When all the changes are complete, enter a hexadecimal number greater than FF (e.g., 100).

The program will then ask if there are any more locations in the current file to be modified. If a "Y" is typed in, the program will ask for another word address, and the process starts all over again. If the response is an "N", the file is re-written onto the disk. The original file is preserved intact, and the updated file is given the same name with a different version number.

Unclassified
MJY-03
26 January 1978
Page 2

This program is designed to modify only those files which are in the form of an EPROM data file, which will subsequently be used by the EPROM Programmer.

If there are any problems or suggestions, I can be reached at extension 2563.


M. J. Young
Advanced Electronic Techniques
Wayland M-9, x2563

cc: V. Follansbee
J. Hepp
G. Sarafinas
R. Smith
G. Works

A-11

Intecolor Utility Routines

INTECOLOR UTILITY ROUTINES

Several utility routines were developed for the Intecolor to help support the IDOS-1 and IDOS-0 programs. Many were already resident in the ISC supplied software package and contained in ROM. Those which were designed by Raytheon were also placed in an EPROM and are contained in locations 1800 through 1FFF (hex). These routines are varied, and are used extensively by the Intecolor software. To aid in using the utilities for future software development, or in modifying existing code, a table of available routines, starting addresses, inputs, outputs, the function of the routine, and register usage has been compiled (see Table A-4).

Some of the utility routines have little or no useful purpose to normal programs. Others, however, (e.g., GNUM, SPNOR, GCMA, etc) are very useful in designing a command parser and other interactive programs. I/O routines are also useful, and are described in detail in the Intecolor Users Manual (reference A-1), in the CPU Operating System section.

INTECOLOP UTILITY ROUTINES

START ADDR	NAME	DESCRIPTION	REGISTER	USAGE
			INPUT	OUTPUT
0105	CI	Input character from console device		A=CHARACTER
0106	PRNTR	Okidata Printer handler (not supported)	A=CHARACTER	
0109	CC	Prints character to console device	A=CHARACTER	
010C	PO	Outputs character to punch device	A=CHARACTER	
010F	LO	Prints character to list device	A=CHARACTER	
0112	RSTRT	Entry into CPU OS breakpoint routine		
0115	BLDAD	Basic LOAD? Command handler		
0118	BSAVE	Basic SAVE Command handler		
011B	CI			
011E	KTST	Checks for key pressed interrupt (old keyboard only)		Z=NO KEY NZ=KEY
0121	CI			
0124	LO			
0127	RECEX	Reenter CRT OS (init)		
012A	OSTR	Outputs string to list device until 239 is encountered. Repeat string supported by 237,N,D1,D2,...Dn,238	HL=ADDRESS OF STRING	HL=END OF STRING + 1 LE=WORKSPACE IF 237...238 USED
012D	CONT	Reenter CPU OS (no init)		
0130	WATS	Wait 0.5 MS/count	A=COUNT	A=0
0133	EXPR	Evaluate CPU OS expression		
0136	LBRT	Output byte to list device in Ascii Hex form	A=BYTE	A=BINARY NISSE C0=NOT HEX INCJ=FOUND
0139	NIBL	Checks byte in A for Ascii hex character, returns binary nibble if found	A=ASCII BYTE	A=10H A=0
013C	CRLF	Outputs a carriage return/line feed to list		
013F	WATL	Wait 20 MS/count	A=COUNT	
0142	CMPHD	Compare DE to HL		
0145	SUBCH	Subtract DE from HL		
0148	SPACES	Ignore spaces	HL=ADDRESS OF STRING	
014B	ERRMSG	FDS Error message handler		
014E	WFI	FDS Write file routine		

MULTICOLOR UTILITIES ROUTINES

START ADDR	NAME	DESCRIPTION	REGISTER USAGE	INPUT	OUTPUT
0151	RZ2	FCS Read file routine	FPR HAS INFO		(C)=FPR B=ERROR CODE
0154	CHDLR	Indirect jump to disk handler	HL=ADDRESS OF FPR HL=ADDRESS OF COMMAND		
0157	PFSPC	FCS Filename parser	B=BYTE COUNT HL=SOURCE ADDRESS DE=DESTINATION ADDR B=DESTINATION ADDR DE=SOURCE ADDRESS HL=DESTINATION ADDR		
015A	MOVH	Move (HL) to (DE)			
0160	MOVH	move (DE) to (HL)			
0163	CHPDH	Compare HL to DE			
0166	LTNR	Ignore letters	HL=ADDRESS OF STRING		(C)=HL<DE HL=FIRST NON-LETTER CHAR Z=END OF LINE
0169	LET	Check byte for Ascii letter	HL=ADDRESS OF BYTE		(C)=LETTER (NC)=NO LETTER
016C	DIG	Check byte for Ascii digit	HL=ADDRESS OF BYTE		(C)=DIGIT FOUND (NC)=NO DIGIT
016F	LDG	Check for letter or digit	HL=ADDRESS OF BYTE		(C)=LETTER OR DIGIT (NC)=NEITHER
0172	GDMA	Get first nonblank char after comma If no comma, go to first nonblank char	HL=ADDRESS OF STRING		HL=FIRST NON BLANK CHAR AFTER COMMA (C)=COMMA FOUND
0175	MSTR	Move string from (HL) to (DE); if non-letter or digit found, move stops, blank fill	HL=SOURCE ADDRESS DE=DEST ADDRESS I=BYTE COUNT		A=#BYTES MOVED HL=HL+A DE=DE+8 A=20H A=17 HL=HL+B B=0 A=LAST CHAR LISTED
0178	PSPAC	Print space			
0179	PCOLM	Print colon			
017E	PSTR	Print space, then print char string	B=CHAR COUNT HL=SOURCE ADDRESS		
0181	PSTR	Same as PSTR, but no space before			
0184	PSBYT	Print space, then list byte in 2 Ascii Hex characters	HL=SOURCE ADDRESS		
0187	PSYT	Same as PSBYT but no space before			

INTECOLOP UTILITY ROUTINES

PAGE 3

START ADDR	NAME	DESCRIPTION	INPUT	REGISTER USAGE	OUTPUT
018A	PS2NUM	Print 2 spaces, then 16-bit number at (HL) in Ascii hex (10 byte, hi byte)	HL=SOURCE ADDRESS		HL=HL+2
018D	PSNUM	Same as PS2NUM, but only 1 space before			
0190	PNUM	Same as PS2NUM, but no space before			
0193	GN1Z	DE=0, then DE=up to 4-digit Hex number after comma or space	HL=SOURCE ADDRESS		DE=NUMBER HL=NEXT NONBLANK CHARACTER DE=DE+NUMBER DE=NUMBER INC=NUMBER FOUND INC=NO NUMBER HL=NEXT CHR AFTER NUMBER DE=DE+NUMBER HL=HL+A HL=HL AND DE HL=-HL HL=NOT HL HL=HL OR DE HL=HL XOR DE HL=HL*(28DE) HL=HL/(28DE) DEHL=HL#DE HL=DE/HL HL=BLOCK COUNT C=LAST BLOCK BYTE COUNT HL=BYTE COUNT
0195	GN1D	Same as GN1Z but de not cleared			
0199	GN2Z	DE=0, then DE=up to 4 digit hex number after blankHL=SOURCE ADDRESS			
019C	GN2D	Same as GN2Z, but DE not cleared			
019F	ADHLA	Add A to HL			
01A2	ANHD	AND HL with DE			
01A5	NEGH	Negate HL			
01A8	NOTH	Complement HL			
01AB	ORHD	OR HL with DE			
01AE	XORHD	XOR HL with DE			
01B1	SHLHD	Shift HL left by DE bits			
01B4	SHRHD	Shift HL right by DE bits			
01B7	MULHD	Multiply DE by HL			
01BA	DIVHD	Divide DE by HL			
01BD	PC2BK	Convert total byte count to number of blocks and last block byte count	HL=BYTE COUNT		
01C0	2K2BC	Convert number of blocks and last block byte count to total byte count	HL=BLOCK COUNT C=LAST BLOCK COUNT		
01C3	BSB01				
01C6	PSB02				
01C9	BSB03				
01CC	BSB04				
01CF	BSB05				
01D2	ESP06				
01D5	BSB07				
01E8	PSB08				
01EB	CRTHD	Init CRT OS entry			
01EE	CRTHD	Reset init (same as ESP06)			
01F1	21VIT				

INTECDOC UTILITY ROUTINES

START ADDR	NAME	DESCRIPTION	REGISTER USAGE	INPUT	OUTPUT
01E4	ERSTR	Basic restart (same as ESC E)			
0400	GETIO	Look for 'IO', set error code	HL=STRING ADDR		(Z)=FOUND (NZ)=NOT FOUND
0403	CKEND	Check for end of line, set syntax error if not	HL=ADDRESS		(Z)=OK (NZ)=NOT EOL
0406	RESET	Resets floppy disks and drivers			(C)=ERROR E=ERROR CODE
0409	FCS	FCS command parser & executor			(C)=ERROR E=ERROR CODE
040C	FCSEM	Process FCS command & process any errors			(C)=ERROR E=ERROR CODE
040F	OPENX	Open file & set up FPB, add user to handler	FPB=INFO		(C)=INVALID DEVICE (Z)=NO DEV, DEFAULT USED
0412	OPEN	Open file	FPB HAS INFO		
0415	CLOSE	Close file	FPB HAS INFO		
0418	PDV	Parse device name	HL=ADDR OF COMMAND		
041B	RWSEQI	Rewind sequential input file			
041E	INSEQD	Initialize sequential output file			
0421	CLSEQD	Close sequential output file			
0424	RBLK	Read block			
0427	RLKI	Read block & increment block number			
042A	WBLK	Write block			
042D	WBLKI	Write block & increment block number			
0430	GTBYT	Get byte			
0433	PTBYT	Put byte			
0436	CAREC	Get ASCII record			
0439	PUREC	Put variable length record			
043C	PTREC	Put unformatted record			
7084	RCMK				
7863	OPDIR	Open directory			
7869	GNDE	Get next directory entry			
7989	READ	Read file 'image'			
79E2	WRITE	Write file 'image'			
:600	CYPER	Set baud rate to 300 and enable Cyber interface Cursor left omitted as Cyber backspace (home), DELETE CTR			A=127

INTEC-200 UTILITY ROUTINES

PAGE 5

START ADDR	NAME	DESCRIPTION	REGISTER USAGE	INPUT	OUTPUT
1803	OCTLZ	returns to caller program Clear DE, then DE=up to 5 digit octal number at (HL)		HL=SOURCE ADDRESS	DE=NUMBER HL=NEXT CHAR AFTER NUMBER ICJ=NUMBER FOUND DE=DE+NUMBER ICJ=ENTRY FOUND ICJ=NOT FOUND HL=HL+1 HL=HL+1 AFTER CORRECT ENTRY
1806	OCTAL	Same as OCTLZ but DE not cleared		HL=TABLE ADDRESS	DE=NUMBER ICJ=NUMBER FOUND DE=DE+NUMBER ICJ=ENTRY FOUND ICJ=NOT FOUND HL=HL+1 HL=HL+1 AFTER CORRECT ENTRY
1809	LOGUP	Searches table at (HL) for word at (DE)		HL=WORD ADDRESS	DE=NUMBER ICJ=NUMBER FOUND DE=DE+NUMBER ICJ=ENTRY FOUND ICJ=NOT FOUND HL=HL+1 HL=HL+1 AFTER CORRECT ENTRY
180C	GNUM	Fetch number in decimal, hex (H), or octal (V)		HL=SOURCE ADDRESS	DE=NUMBER ICJ=NUMBER FOUND DE=DE+NUMBER ICJ=ENTRY FOUND ICJ=NOT FOUND HL=HL+1 HL=HL+1 AFTER CORRECT ENTRY
180F	DECAZ	DE=0, then get decimal number into DE		HL=SOURCE ADDRESS	DE=NUMBER ICJ=NUMBER FOUND DE=DE+NUMBER ICJ=ENTRY FOUND ICJ=NOT FOUND HL=HL+1 HL=HL+1 AFTER CORRECT ENTRY
1812	DECHL	Same as DECAZ but DE not cleared		HL=SOURCE ADDRESS	DE=NUMBER ICJ=NUMBER FOUND DE=DE+NUMBER ICJ=ENTRY FOUND ICJ=NOT FOUND HL=HL+1 HL=HL+1 AFTER CORRECT ENTRY
1815	BRDEC	Output number in A to memory as a 3-digit decimal number		HL=DESTINATION ADDR A=BINARY NUMBER	DE=NUMBER ICJ=NUMBER FOUND DE=DE+NUMBER ICJ=ENTRY FOUND ICJ=NOT FOUND HL=HL+1 HL=HL+1 AFTER CORRECT ENTRY
1818	FLPTT	Fetch floating point number		HL=ADDRESS OF NUMBER	DE=NUMBER ICJ=NUMBER FOUND DE=DE+NUMBER ICJ=ENTRY FOUND ICJ=NOT FOUND HL=HL+1 HL=HL+1 AFTER CORRECT ENTRY
181B	SETOC	Set radix flag at A000 to octal		HL=SOURCE ADDRESS	DE=NUMBER ICJ=NUMBER FOUND DE=DE+NUMBER ICJ=ENTRY FOUND ICJ=NOT FOUND HL=HL+1 HL=HL+1 AFTER CORRECT ENTRY
181E	PS216	print 2 spaces, then 16-bit number in octal or hex, depending on flag at A000H (0=HEX, 1=OCTAL) (high byte, low byte)		HL=SOURCE ADDRESS	DE=NUMBER ICJ=NUMBER FOUND DE=DE+NUMBER ICJ=ENTRY FOUND ICJ=NOT FOUND HL=HL+1 HL=HL+1 AFTER CORRECT ENTRY
1821	PS16	Same as PS216, but only one space before		HL=SOURCE ADDRESS	DE=NUMBER ICJ=NUMBER FOUND DE=DE+NUMBER ICJ=ENTRY FOUND ICJ=NOT FOUND HL=HL+1 HL=HL+1 AFTER CORRECT ENTRY
1824	P16	Same as PS216, but no space before		HL=SOURCE ADDRESS	DE=NUMBER ICJ=NUMBER FOUND DE=DE+NUMBER ICJ=ENTRY FOUND ICJ=NOT FOUND HL=HL+1 HL=HL+1 AFTER CORRECT ENTRY
1827	PNUM0	Output number in octal or hex, depending on flag at A000H (0=HEX, 1=OCTAL) (low byte, high byte)		HL=SOURCE ADDRESS	DE=NUMBER ICJ=NUMBER FOUND DE=DE+NUMBER ICJ=ENTRY FOUND ICJ=NOT FOUND HL=HL+1 HL=HL+1 AFTER CORRECT ENTRY
182A	PRTCN	Turn on printer, turn off CRT		HL=SOURCE ADDRESS	DE=NUMBER ICJ=NUMBER FOUND DE=DE+NUMBER ICJ=ENTRY FOUND ICJ=NOT FOUND HL=HL+1 HL=HL+1 AFTER CORRECT ENTRY
182D	PRTCO	PRTCO		HL=SOURCE ADDRESS	DE=NUMBER ICJ=NUMBER FOUND DE=DE+NUMBER ICJ=ENTRY FOUND ICJ=NOT FOUND HL=HL+1 HL=HL+1 AFTER CORRECT ENTRY
1830	GFLT	Get floating point number, if no fraction, FE=0		HL=SOURCE ADDRESS	DE=NUMBER ICJ=NUMBER FOUND DE=DE+NUMBER ICJ=ENTRY FOUND ICJ=NOT FOUND HL=HL+1 HL=HL+1 AFTER CORRECT ENTRY
1833	PRTCO	Last 16-bit number in decimal, leftmost sign, leading 0s blanked		HL=SOURCE ADDRESS	DE=NUMBER ICJ=NUMBER FOUND DE=DE+NUMBER ICJ=ENTRY FOUND ICJ=NOT FOUND HL=HL+1 HL=HL+1 AFTER CORRECT ENTRY
1836	BN-EX	Convert binary number in A to Hex and store in memory		HL=SOURCE ADDRESS	DE=NUMBER ICJ=NUMBER FOUND DE=DE+NUMBER ICJ=ENTRY FOUND ICJ=NOT FOUND HL=HL+1 HL=HL+1 AFTER CORRECT ENTRY
1839	PS77	Output string till null found		HL=SOURCE ADDRESS	DE=NUMBER ICJ=NUMBER FOUND DE=DE+NUMBER ICJ=ENTRY FOUND ICJ=NOT FOUND HL=HL+1 HL=HL+1 AFTER CORRECT ENTRY
183C	PRTCN	Display to bit section of memory		HL=SOURCE ADDRESS	DE=NUMBER ICJ=NUMBER FOUND DE=DE+NUMBER ICJ=ENTRY FOUND ICJ=NOT FOUND HL=HL+1 HL=HL+1 AFTER CORRECT ENTRY

INTECOLP UTILITY ROUTINES

START ADDR	NAME	DESCRIPTION	REGISTER USAGE	INPUT	OUTPUT
				C=WORD COUNT DE=DISPLAY ADDRESS HL=DATA ADDRESS C=0	
183F	LOAD	Load hex MAC80 file from Cyber to memory			
1842	POS	File in standard Intel Hexadecimal format			
1845	R1	Output string to punch device			
		237 steps transmission			
1848	IDCS	Input character from RS-232 port 1 and convert to binary number			
		Common Element bus driver for Terminal Interface Element (TIE)			
					A=NUMBER A=ERROR STATUS (C)=ERROR (NC)=NO ERROR
		0 - WRITE Write to XMIT RAM			
		1 - READ Read from Receiver RAM			
		2 - WRTR Write reset			
		3 - READR Read reset			
		4 - SELBUSA Select Bus A			
		5 - SELBUSB Select Bus B			
		6 - XMIT Transmit to bus			
		7 - REQICS Request I/O status			
		8 - SETVAD Set virtual address			
		9 - POLINT Check interrupt lines			
		10 - LRAMAD Load receiver ram address			
		11 - SEND Write to Xmit ram and Xmit, return status in a			
		12 - DISACK Disable Xmit, receiver ACK			
		13 - EMACK Enable xmit, receiver ACK			
		14 - MSPEN 16 bit transfers			
		15 - MSPDIS 8 bit transfers, 0 fill high byte			
184B	CHEX	Input Ascii Hex numbers from keyboard			
184E	SCRLO	List character, with window scrolling feature enabled			
		Parameter Block Format:			
		WD0 - Starting line number (1 byte)			
		WD1 - Starting character in line (1 byte)			
		WD2 - Number of lines (1 byte)			
		WD3 - Number of characters in line (1 byte)			
					A=RETURNED STATUS (Z)=STATE OF INTERRUPTS A=RETURNED STATUS HL=NUMBER HL=ADDRESS OF PARAM

INTERCOLOR SYSTEM ROUTINES

START ADDR	NAME	DESCRIPTION	REGISTER USAGE	INPUT	OUTPUT
1851	SCRSO	W24 - Current line count (1 byte) List string, with window scrolling feature enabled HL=ADDRESS OF PARM		BLOCK	
1854	EWINDO	Erase window of display		DE=ADDRESS OF STRING HL=ADDRESS OF PARM	
1857	SVCSR	Save current location of visible cursor Any number of cursor saves may be performed. Location will be placed on a stack		BLOCK NONE	
185A	RSCSR	Restore visible cursor location		NONE	
185D	SCROLL	Scroll window of display		HL=ADDRESS OF PARM BLOCK	
1860	SETHX	Set radix flag at A000 to hexadecimal			A=0
1863	PRTOFF	Turn off IDS printer			



FORM 10-0557 (9-65) GOWD

DIVISION EQUIPMENT
Operation EDL - Wayland
Department Advanced Development Laboratory

Classification Unclassified
Contract No. IDP 79D-320
Distribution EDL-94 S,M Code
File No. EM79-0498
Memo No. RJB-185
Date 24 August 1979

To File
From R. J. Bonneau
Subject FTSR CE Diskette Utility Programs

I. Introduction

Two new BASIC utility programs (PGCOPY, PGCOMP) have been installed on the Intecolor Support Program disks (13 and 13A) at the Sudbury Test Site. The purposes of these programs are to facilitate the copy of CE disk pages from one disk to another, and to perform a page comparison between two CE disks. This memo briefly describes the operational use of these programs, and other CE diskette maintenance programs.

II. PGCOPY - Copy Pages

PGCOPY is a program to automate the process of copying a set of CE pages (i. e. , files with name of the form Page. nnn) from drive 1 to drive 0. The program accepts up to 20 input page numbers (3 decimal digits), then automatically copies these pages from Drive 1 to Drive 0. The program is invoked by executing, in BASIC, the following two commands (with the program disk in drive 0):

LOAD?"PGCOPY"

RUN

After RUN is executed, the user should insert in drive 1 the from diskette, remove the utility disk from drive 0, and insert the to disk and then proceed to respond to the program request:

ENTER 3 DIGIT PAGE NUMBER?

The user types in a 3-digit number (e. g. , 031 or +27, or 005) and hits the return key. The program continues to request pages, up to 20, until the sequence *** is entered for a page number. This terminates the entry phase and begins the copying phase. When all the requested pages have been copied, the entire sequence may be repeated. Note that this program copies only the highest numbered version of the page file.

III. PGCOMP - Compare CE Pages Program

PGCOMP performs a word by word comparison of CE page files on two diskettes and displays any discrepancies. This program is useful for verifying master CE disks versus working CE disks.

The program is invoked by inserting the support programs disk (13 or 13A) onto drive 0, entering BASIC (ESC W and return), then entering:

LOAD?"PGCOMP" (return)

RUN (return)

At this point, the utility diskette should be removed from drive 0, replaced with one of the two CE disks, with the other disk going to drive 1. Then the program requests:

ENTER NAME OF FILE?

The user enters a file name (of the form PAGE.nnn) and hits return key. (The user may optionally specify version number; vv). The program now reads this file, from both disks and does a byte by byte comparison of the 480 bytes on a page file. Any discrepancies are listed on the display in the form:

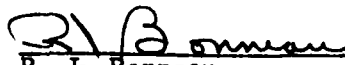
BYTE # xx DRIVE 0 yy DRIVE 1 zz

After the entire file has been checked, the total number of discrepancies is output and the program loops back to request another file name spec.

IV. Suggested Improvements for Support Programs

These two programs, along with the LOADCE, GET, RETURN, and MERGE programs, constitute CE diskette file maintenance programs. There are a number of areas in which we can improve these support tools.

- A. PGCOPY should be updated to facilitate copying of a complete task of pages from one disk to another. This would involve accessing a load map page and use it to drive the page number inputs.
- B. PGCOMP should also be updated to include comparison of a complete task from one disk to another.
- C. LOADCE, GET and RETURN programs need updates to extend the page numbering limit from 99 to around 150 or more.
- D. An additional program to enable the automatic printing of a task (load map page and all its object pages) is also very desirable for documentation phases.
- E. Similarly, a program to analyze a CE diskette's directory and produce a listing of page allocation on a task by task basis would be very useful for maintaining automatic inventory of CE diskettes.
- F. Finally, it would be most productive to gather together, into one master program, all of the CE diskette maintenance programs and provide a menu approach to user interaction. This can be done quite easily inasmuch as the Intecolor BASIC supports executing of the LOAD? and RUN commands as BASIC commands.


R. J. Bonneau
Advanced Electronic Techniques
Wayland Box M9, x5171

RJB/lc

Attachment: PGCOPY Listing
PGCOMP Listing

```

50 CLEAR 1000
10 DIM A$(20)
90 PRINT "ENTER PAGE NUMBER(S) AS 3 DIGITS - TERMINATE WITH ***"
95 PRINT "THE PAGE WILL BE COPIED FROM DRIVE 1 TO DRIVE 0"
5 FOR I=1 TO 20
100 INPUT A$(I)
105 IF A$(I)="***" THEN 107
105 PRINT I
107 FOR J=1 TO I-1
110 PLOT 27:PLOT 4
110 B$="PAGE."+A$(J)
110 PRINT "COPY 1:"+B$+" TO 0:"
110 PLOT 27:PLOT 27
115 NEXT J
120 GOTO 90

```

```

100 PRINT "ENTER NAME OF FILE"; INPUT A$
105 C=0:REM NUMBER OF DISCREPANCIES
110 PLOT 27:PLOT 4
120 PRINT "LOA 0:" + A$ + " B000"
125 PLOT 27:PLOT 27
135 PLOT 27:PLOT 4
140 PRINT "LOA 1:" + A$ + " C000"
150 PLOT 27:PLOT 27
200 FOR I=0 TO 479
210 X = PEEK (-20480+I)
220 Y = PEEK (-16384+I)
230 IF X=Y THEN 240
232 C=C+1
235 PRINT "BYTE # ";I,TAB(15);"DRIVE 0 ";X,TAB(30);" DRIVE 1 ";Y
240 NEXT I
250 PRINT "NUMBER OF DISCREPANCIES = ";C
300 GOTO 100

```

REFERENCES

- A-1. Intecolor 8001 Users Manual, Intelligent Systems Corp.,
Norcross, Georgia.

APPENDIX B

Subroutines and Data Structures in FTWRP

- B-1 DOS-0 Data Structures
- B-2 IDOS-0 Hierarchy and Subroutine Usage
- B-3 IDOS-1 Subroutines
- B-4 IOC Continuous Input Mode Subroutines

B-1

DOS-0 Data Structures

LEVEL 0 SYSTEM DATA STRUCTURES

	<u>STRUCTURE</u>	<u>USED BY</u>
1.	INPUT/OUTPUT QUEUES	DOS-0
2.	FREE ENTRY STACK	DOS-0
3.	LEVEL 0 REQUEST STACK	DOS-0
4.	SYSTEM REQUEST DATA PACKETS	USER PROGRAM
	4-a. PHILOSOPHY OF USER LEVEL I/O	
	4-b. SYSTEM I/O PACKET STACK	DOS-0
5.	TASK LOAD MAP PAGE	DOS-0
	5-a. COMMON MEMORY DIRECTORY PAGE	
6.	SYSTEM STATUS BLOCK	DOS-0 & DOS-1
	6-a. PHILOSOPHY OF TASK SUSPENSION AND	
7.	CONFIGURATION TABLE	DOS-0
8.	LOGICAL DEVICE LIST	USER PROGRAM
9.	TASK PROLOGUE AREA	USER PROGRAM
	9-a. UNSOLICITED INPUT - BUFFER FORMATS	USER PROGRAM
10.	EXECUTIVE MESSAGE FORMATS	DOS-0
	10-a. DOS-1 TO DOS-0 MESSAGES	
	10-b. DOS-0 TO DOS-1 MESSAGES	
	10-c. DOS-0 TO TRACE DEVICE	
11.	I/O HARDWARE STATUS WORD FORMAT	USER & DOS-0
12.	DOS-1 FAULT MESSAGES/FAULT BITS	OPERATOR & DOS-1
13.	CE PROGRAM STATUS WORD	DOS-0
14.	SYSTEM STATUS RETURN MESSAGE FORMAT	DOS-0 & DOS-1

UPDATE SUMMARY FOR REVISION A

Section 4 System Requests

- Corrected specification of multiple input request parameter words.
- Added three more system requests:
 - 7 - Data recording request
 - 8 - Update data recording control words
 - 9 - Modify virtual address

Section 5 Task Load Map Page

- Included definition for an entry to be used as a patch page.

Section 6 Common Element System Status Block

- Defined Bit 9 of Word 0 as Bus Alternation Indicator

Section 9 CE Task Prologue

- Added separate PSW values for unsolicited input and clock interrupt entries
- Added modify virtual address indicator word
- Added entry addresses and PSW's of direct I/O of message codes 5, 6, 10, 11
- Added data recording control words.

Section 10 DOS-1 to DOS-0 Messages

- Added bus control bits to configuration update message
- Added two new types:
 - 6 - Update virtual address modification control word
 - 7 - Update data recording control words

Section 10-B DOS-0 to DOS-1 Messages

- Schedule task request uses indicator word of 8

Section 12 Fault Messages/Fault Bits

- Added four more faults; three for data recording, one for PROM checksum testing.

UPDATES TO THE SYSTEM DATA STRUCTURES DEFINITIONS

Section 4 System Requests

- Added a new action to the task directives - Swap Tasks
- Added a new request (6) for registering a user detected fault into the task status return message. Allowed up to 16 different fault indicators.

Section 8 CE Logical Device List

- Added Device 9 - CE Diagnostic Task. This task is loaded by each CE prior to tactical operations and may be loaded at a later time for detailed testing. The task number is 74 (octal).

Section 10 Executive Messages

Section 10A DOS-1 to DOS-0 Messages

- Added a new message type (4) - Modify Memory. This message type permits real-time patching of user programs with the aid of DOS-0. The message specified the number of patches, followed by the address-new value pair for each patch.

Section 10B DOS-0 to DOS-1 Messages

- Added the swap tasks message to the task directives type. The information supplied consists of the virtual addresses to be swapped along with starting addresses of each task when started.

1. INPUT/OUTPUT QUEUES (I/OQ)

- Separate double-linked lists for the input requests and the output requests.
- Used to keep track of pending or ongoing I/O activities of the CE.
- Each entry consists of 24 words in the format below.

<u>Word</u>	<u>Mnemonic</u>	<u>Description</u>
0	IOFP	Forward link (0 if at end of list)
1	IOBP	Backward link (0 if at head of list)
2	IOPA	Packet address of request
3	IOHW	Header word list address (points to IONMHD)
4	IOCWC	Current word count
5	IOCBA	Current buffer address
6	IOSTAT	Entry status word
7	IOOPTS	Entry options word (see packet option word)
8	IOCMPG	Common memory page number
9	IOEA	Entry address for associated I/O request
10	IOVA	Virtual address of eventual destination
L1	IOVANM	Index of virtual address in header word list
12	IONMHD	Number of header words in following list
13-20	IOHDRS	Header words
21	IOIDLT	Multiply input request - Address Delta
22	IOICNT	Multiply input request - Number of Requests
23	- - -	Spare Word

- The status word (IOSTAT) contains the following information:

<u>Bit Number</u>	<u>Description</u>
15	Entry in use
12	I/O request completed
REST	Undefined

1. INPUT/OUTPUT QUEUES (I/OQ) (Continued)

- Each queue has a number of pointer and counting variables

<u>Variable</u>	<u>Usage</u>
ITOP, OTOP	Indicates I/O entry at the top (front) of queue
IBOT, OBOT	Indicates I/O entry at the bottom (tail) of queue
ICURNT, OCURNT	Indicates I/O entry currently being processed
ICOUNT, OCOUNT	Number of entries currently in the queue

- Currently, 20 entries available for use for both input and output.

2. FREE ENTRY STACK (FS)

- This stack contains addresses of available I/O queue entries.
- As I/O queue entries are requested (through the routine DOPOPF) and returned (through DOPIHF), the free stack grows and shrinks.
- There are several pointer and counting variables related to FS:

<u>Variable</u>	<u>Contents</u>
FST	Free Stack Top
FSB	Free Stack Bottom
FSC	Free Stack Current
FCOUNT	Count of free entries

3. LEVEL 0 REQUEST STACK (LORS)

- Used by level 0 to respond to requests from higher level (interrupt) routines.
- Each entry in stack contains four words:

<u>Word</u>	<u>Contents</u>
0	Request Number - 1 \Rightarrow No request
1 - 3	Supplemental data

- Typical requests currently envisioned are:
 - Load user task (Request Number 0)
Data = User task load map page number
 - Start user task (Request Number 1)
Data = Task number, start address (or ϕ)
 - Send fault message (Request Number 2)
Data = Fault identifier, 2 words of fault data.
See Section 12 for fault identifiers and message data.
- Currently, five entries available
- There are several pointer variables:

<u>Variable</u>	<u>Usage</u>
LORST	Top of LORS
LORSB	Bottom of LORS
LORSC	Current entry of LORS

4. SYSTEM REQUESTS - DOS-~~Ø~~

<u>Request #</u>	<u>Data Packet Format</u>	
0 Write	Word 0	Status Word
	Bit 15	Request Queued
1 Read	Bit 12	I/O Completed
	Bits 11, 10	11 - I/O Error
		10 - Data Management Error
	Bits 7 - 0	11 - Status Byte
		10 - Code
	Word 1	Device Number (or Header List Address)
	Word 2	Word Count
	Word 3	Buffer Address (absolute address)
	Word 4	Options
	Bit 15	Executive Message (output only) - Uses Word 7
	Bit 14	No Retry on Recoverable Errors
	Bit 13	Device = Header List Address (output only)
	Bit 12	Scatter Read (input) (uses Bits 7 - 0)
	Bit 11	System Packet (Executive only)
	Bit 10	Associated Input Request (Exec only)
	Bit 9	Common Memory Request
	Bit 8	Multiple Input Request
	Bit 7	0 → Complex Scatter Read 1 → Real Scatter Read
	Bits 6 - 0	Scatter Read Interval (1 - 127)
	Word 5	Common Memory Page # or Multiple Input Request Parameter
	Bits 15 - 8	Number of Requests
	Bits 7 - 0	Address Delta
	Word 6	Associated Input Entry Address (DOS-Ø use only)
	Word 7	Executive Header Skeleton (output only) (Must contain message code - may contain destination V A)

4. SYSTEM REQUESTS - DOS-Ø (Continued)

<u>Request #</u>	<u>Data Packet Format</u>	
2 Trace Update	Word 0	Bit 15 Clear - Stop tracing Set = Start tracing Bits 7 - 0 Trace frequency interval
3 Unsolicited Input Update	Word 0	New Options Word (see Prologue for Definition)
	Word 1	New Data Buffer Address
	Word 2	New Header Buffer Address
		} Relative to DORG Ø
4 Clock Inter- rupt Update	Word 0	New Options Word
	Words 1 - 2	New Clock Period
5 DOS-1 Task Directives	Word 0	Directive Types
		1 = Schedule task
		2 = Suspend task
		3 = Resume task
		4 = Abort task
		5 = Swap tasks
	For Types 1, 2, 3, 4	
	Word 1	Task number
	Word 2	Starting Address for Task Execution (If 0, use address in prologue)
	For Type 5 (Swap Tasks)	
	Word 1	Bits 15 - 8 Virtual Address 1 of Swap Bits 7 - 0 Virtual Address 2 of Swap
	Word 2	Starting Address for Virtual Address 1
	Word 3	Starting Address for Virtual Address 2
6 Register User Fault	Word 0	User Fault Number (0 to 15)

4. SYSTEM REQUESTS - DOS- ϕ (Continued)

<u>Request #</u>	<u>Data Packet Format</u>
7 Data Recording Request	<p>Word 0 Extraction Point Number</p> <p>Word 1 Number of Subrecords</p> <p>Each Subrecord has Format:</p> <p>Word 2 Number of Words (Subrecord 1)</p> <p>Word 3 Absolute Address of Data to be Recorded</p> <p>Word 4 Number of Words (Subrecord 2)</p> <p>Word 5 Absolute Address of Data to be Recorded</p> <p style="text-align: center;">⋮ ⋮</p>
8 Update Recording Control Words	<p>Word 0 Logical Device Number (see Section 8)</p> <p>Word 1 Recording On Flags</p> <p>Word 2 Recording Off Flags</p>
9 Modify Virtual Address	Data Packet Address = New Virtual Address

4-A. PHILOSOPHY OF USER LEVEL I/O

When a user makes a request for I/O, the status word of the user packet (Word 0) must be set to 0. After the request is made, the status word must be checked in order to verify that the request was queued. This is done by testing Bit 15; if set, the request has been queued and the I/O will be attempted. If not, DOS-Ø was unable to find space in its data base for the I/O request.

Once a request has been queued, the status may be checked by interrogating the I/O complete bit in the status word (Bit 12). Until all of the requested I/O has been completed or an error has occurred, this bit remains 0. Thus the user should periodically check this bit. Once it is set, the determination of correct or incorrect termination of the I/O request is determined through the two error bits, 11 and 10.

Bit 11 set indicates an error has occurred in the attempt to transmit or receive data. If Bit 11 is set, the bottom byte of the status word (Bits 7 - 0) contains the I/O status word from which can be obtained the actual error. (see Section 11).

Bit 10 set indicates an error has occurred in DOS-Ø's attempt to manage the data represented by this request. In this case, the bottom byte contains a code indicating which type of error occurred; e. g. , word count too large, etc.

If neither error bit is set, the I/O request has been completed as requested and the user may now reuse the packet for other requests.

If a user request for input generates an output request, and an error occurs in that output processing, both the output request and input request packets are set to indicate an error, which would be the same error in both packets.

4-A. PHILOSOPHY OF USER LEVEL I/O (Continued)

Option Bits

Executive Message (Bit 15) (Output Only)

Used for output only. Permits the modification of the header list of the device selected by OR'ing in the executive header word (Word 7 of packet) into the destination header word. This skeleton must contain a valid message code and may or may not contain a destination address, depending on the device.

No Retry on Error (Bit 14) (Output Only)

If this bit is set and a recoverable error is encountered (e.g., bus busy or receiver busy), the system will not retry the I/O.

Header List Supplied (Bit 13) (Output Only)

Use of this option directs DOS- ϕ to bypass the device number to header list translation phase of I/O processing by using the header list address supplied in the device word field of the user packet. Use of this option requires that both the destination field and the message code field appear in the header list.

Scatter Read (Bit 12) (Input Only)

If this bit is set, the operating system request performs a scatter read of data, which each word read being separated from the next by a count equal to Bits 6 - 0 of the options word. For example, to scatter read into every 9th word, set Bit 12 and store a 9 into the lower byte. Bit 7 is then used to indicate real (set) or complex (clear) read, i.e., complex reads two words, skip, read two word, etc.

System Packet (Bit 11)

When this bit is set, the DOS- ϕ system has invoked a packet from the packet stack. In the case of I/O message completion, the packet must be returned to the packet stack.

4-A. PHILOSOPHY OF USER LEVEL I/O (Continued)

Associated Input Request (Bit 10) (Output Only)

When set, this bit indicates that the current output request was generated due to an input request to a common memory or a non-bus-extender IOC. In this case, the PAEA field of the I/O packet must contain the I/O queue entry address for the associated input request.

Common Memory Request (Bit 9)

If set, the bit indicates that the CM page field of the packet contain page number to be fetched or stored.

Multiple Input Requests (Bit 8)

Automatically regenerates the specified number of input requests with each input buffer address incremented by specified amount. Uses Word 5 for number of requests and address delta.

Common Memory I/O Requests

Output Requests

Device # (18, 19) → Device Field (Word 1)

Set Bit 9 in Options Word (CM Request) (Word 4)

Set Page # → CM Page # Field (Word 5)

Data in Data Buffer Address → Buffer Address (Word 3)

Word Count 240 (Word 2)

Input Requests

Device # (18, 19) → Device Field (Word 1)

Set Bit 9 in Options Word (CM Request) (Word 4)

Set Page # → CM Page # Field (Word 5)

Buffer Area → Buffer Address (Word 3)

Desired # Words → Word Count (Word 2)

4-B. SYSTEM I/O PACKETS AREA (SYSPA)

- Composed of five entries pointing to a packet plus 5 words of message area (SYSPA)
- Stack used to point to available packets (PASTK)
- Pointer into stack for currently available packet (PASTKC)
- Two routines to pop and push packets
- Counter, to indicate number of available packets (PASTCT)
- Used by system routines for fault messages and DOS-1 messages
- Buffer Address Field points to message area of the entry

5. TASK LOAD MAP PAGE

- Used by the DOS-0 loader (DOLOAD) to control the loading of user programs
- Each user task is broken down into segments which represent up to 240 contiguous words of the task.
- Header words of the load map page tell the type of load map (currently always zero) the number of segments (1 to 59), an identifier (i. e. , task number 27).
- Each segment is specified by 4 words:
 - Common memory page number for the body of the segment
 - Start address of where to load body
 - Number of words to load
 - Checksum of words to be loaded
- Format is given on the next page
- Number of words to load entry may contain Bit 15 set. This indicates a relocatable load of a data segment. DOS-0 responds by loading the segment using the start address specified plus the start address of global data.
- A Load Map Page entry consisting of a stack address and word count both = 0 indicates a patch page and will be handled by the loader as a patch page using format of executive message type 5 (see Section 10).

	Word 0	Word 1	Word 2	Word 3
Header	Map Type	Number of Segments	Identifier	Spare
Segment 1	Page No.	Start Address	Number of Segments	Checksum
Segment 2	"	"	"	"
Segment 3				
Segment 4				
•				
•				
•				
			•	
			•	
Segment 59				

Load Map Format

5-A. COMMON MEMORY DIRECTORY PAGE

- For Common Memories containing stored programs, the directory resides on page 3.
- Two words per entry, one entry per possible virtual address.
- Entry contains an entry status word and a load map page number.
- If entry status word equals the virtual address, then a task with the specified virtual address may be loaded using the load map page number.
- Format:

	<u>Word 0</u>	<u>Word 1</u>
VA 0		
VA 1	0 or 1	Load Map Page Number for Task 1
.	.	.
.	.	.
.	.	.
VA '77	0 or '77	Load Map Page Number for Task '77

6. COMMON ELEMENT SYSTEM STATUS BLOCK (SYSTAT)

- Four word block of summary system status data to be sent to DOS-1 upon status request.
- Word 0 gives state of CE:

<u>Bit</u>	<u>Meaning</u>
15	Initializing
14	Loading a task
13	Task loaded and not executed
12	Task in execution
11	Idle
10	Task Suspended (see next page)
9	Bus Alternation Indicator (0 = Disabled, 1 = Enabled)
8	Summary Error Bit - (If set, error bit(s) set in words 2, 3, or 4)
7-6	01 Card Type = CE
5-0	Virtual Address

- Words 1, 2 describe a number of possible DOS-Ø error situations: where word 1 contains fatal errors and word 2, non-fatal errors. See Section 12 for assignment of error indications.
- Word 3 - reserved for User Task Fault Indicator from 0 - 15.

6-A. PHILOSOPHY OF TASK SUSPENSION & RESUMPTION IN DOS-0

Suspension consists of a loop executed in the Input Handler of DOS-0. The loop is terminated by the receipt of an Executive message requesting either task resume or task abort.

While a task is suspended, no output messages are transmitted; input messages are accepted and processed. Any output complete interrupts received will be processed, but no new messages will be initiated. Also, while a task is suspended, user clock interrupts are not honored.

Task resumption is caused by the receipt of the Executive message to resume the task. In detail, the input pending handler clears the suspended bit in the CE status word, upon which an "earlier" instance of the handler is polling. Before this level is exited, the output routine is called to initiate any output which may have been queued while in the suspended state.

A task abort request is processed by setting up information in the level 0 request stack (10RS) and performing a drop to level 0 to send an error message to DOS-1 and enter into the idle loop awaiting further directions from DOS-1.

7. CONFIGURATION TABLE (CONFIG)

- Contains header lists and supplementary information pertaining to the translation of logical device numbers into virtual addresses.
- For each device in the system, this table contains a 12-word entry in the following format:

<u>Word Number</u>	<u>Mnemonic</u>	<u>Contents</u>
0	VA	Virtual address of the final element designated as the logical device.
1	VAIN	Index into the header list for the header list for the header word containing the virtual address of word 0; i. e. the index to the destination header word. Value lies between 1 and 8.
2	NIID	Number of header words in the following list
3	IHDRS	1 to 8 header which form the header list used for all communications to the specified logical device.
11	- - -	Spare word

- Part of this table is set with default device assignments at start-up time by DOS-0; but may be updated with new configuration data received from DOS-1. Currently, the CONFIG table is initialized for the following logical devices at start-up:

0	System Load Device
1	Bootstrap Load Device
2	System Device
3	Operator and Trace Device (used for pages 0, 1, 2)
4	Fault Display Device

8. COMMON ELEMENT LOGICAL DEVICE LIST

<u>Device #</u>	<u>Description</u>	<u>Seek Igloo Demonstration</u>
0	Program Load Device	'40 through '54
1	Bootstrap Load Device	'17-Device 1-Port D
2	System Device (DOS-1)	'77
3	Operator/Trace Device (Pages 0, 1, 2)	'40 through '54
4	Fault Display Device Page 1	'40 through '54
5	General Device - Master Cluster	0
6	General Device - Slave Cluster 1	N/A
7	General Device - Slave Cluster 2	N/A
8	General Device - Slave Cluster 3	N/A
9	CE Diagnostic Task	'74
10	Radar Data Source	'50 - Device 1-Port C
11	Radar Data Collector	'60
12	Radar Data Processor	'61
13	Radar Data Post Processor	'75
14		
15		
16		
17		
18	Clutter Map - Area 1	'44
19	Clutter Map - Area 2	'45

Also, '73 - intermediate virtual address used during task swap.

9. USER TASK PROLOGUE AREA

- Occupies the first 256 words (octal 400) of the RAM area.
- Supplies information for starting user execution at various entry points
- Maintains the status of task execution
- Format as on next page
- Must be assembled into every program to be run on a CE.

9. CE TASK PROLOGUE (Continued)

<u>Word Number</u>	<u>Contents</u>
0	Task Number [*]
1	Initialization Entry Address [*]
2	Starting Address - Initial Load [*]
3	Unsolicited Input Entry Address [*]
4	Clock Interrupt Entry Address [*]
5	Reconfiguration Entry Address [*]
6	Starting S Value ^{††}
7	Socket Address ^{††}
'10	Global Data Size [*]
'11 - '27	Base Register Values ^{*††}
'30	Unsolicited Input Options ^{*†}
	Bit 15 - Accept Data
	Bit 14
	• Set: Data + Headers → Data Buffer
	• Clear: { Data → Data Buffer Headers → Header Buffer
'31	Data Buffer Address ^{*†}
'32	Header List Buffer Address ^{*†} } Relative to DORG \emptyset
'33	Unused
'34	Clock Option ^{*†} { Bit 15 Set - Clock Interrupt Desired Bits 7 - 0 - Clock Interrupt Frequency
'35 - '36	Clock Period ^{*†} (LSB approximately 2 ms)

(Continued on Next Page)

* Set by User at Assembly Time

† Modified by DOS- \emptyset at User Request

†† Modified by DOS- \emptyset for Operating System Usage

9. CE TASK PROLOGUE (Continued)

<u>Word Number</u>	<u>Contents</u>
'37	Trace Indicator ^{*†} { Bit 15 - Set - Start Trace Clear - Stop Trace Bits 7-0 - Trace Frequency
'40	Modify Virtual Address Indicator Bit 15 = 1 Modify Permitted = 0 No Modify Permitted
'41	PSW Values for Initialization Entry
'42	PSW Values for Starting Address
'43	PSW Values for Unsolicited Input Entry
'44	PSW Values for Clock Interrupt Entry
'45	PSW Values for Reconfiguration Entry
'46 - '47	Unused
'50	Direct I/O Entry Address Message Code 5*
'51	Direct I/O Entry Address Message Code 6*
'52	Direct I/O Entry Address Message Code 10*
'53	Direct I/O Entry Address Message Code 11*
'54	Direct I/O PSW Value Message Code 5
'55	Direct I/O PSW Value Message Code 6
'56	Direct I/O PSW Value Message Code 10
'57	Direct I/O PSW Value Message Code 11
'60 - '77	Trap Locations [*] /Return Addresses ^{††}
'100	Data Recording - Base Extraction Point Number ^{*††}
'101	Data Recording - On/Off Flag Word ^{*††}
'102 - '377	Reserved for Expansion

Must be \emptyset If Not in Use
Must be 2 If Entry in Use

* Set by User at Assembly Time

† Modified by DOS- \emptyset at User Request

†† Modified by DOS- \emptyset for Operating System Usage

9-A. UNSOLICITED INPUT BUFFER FORMATS

- Unsolicited input messages are available to the user task in two portions; the header area and the message area. Below, we present the structure of these two areas.

HEADER BUFFER:

<u>Word Number</u>	<u>Designation</u>	<u>Description</u>
0	N	Number of header words (1 or 8)
1 - N	H_n	Received header words
N + 1	M	Message word count

MESSAGE BUFFER:

0 - M - 1 Words of the message

- The user may request these buffers to be stored in separate areas or to be located in one area, in which case the header buffer preceded the message buffer.
- Because of the unknown nature of unsolicited input, the user should reserve ten words for the header buffer area, and 256 words for the message buffer area.
- The user has the option of altering the unsolicited input options in real time through a system request (number 3 - see Section 4).

10. EXECUTIVE MESSAGES

10-A. DOS-1 to DOS-0 Messages

- All use a message code of 14.
- All the DOS-1 to DOS-0 messages have the following format:
 Word 0 Message type
 Words 1 - n Body of the message
- The various message types and the structure of the body of the message is given below:

<u>Message Type</u>	<u>Format of the Message Body</u>		
0 Load Task	Word 1		Load Map Page Number
1 Start Task	Word 1		Task Number
	Word 2		Starting Address or 0
2 Configuration Data Update	Word 1	k	Number of Configuration Items plus Bus Control Word
		Bit 15	0 = No alternating 1 = Alternating
		Bit 14	0 = Bus A 1 = Bus B
		Bits 13 - 0	Number of items
	Word 2	1	Number of Words for Item 1 including this word count
	Word 3	d	Device Number
	Word 4	VA	Virtual Address of Actual Card
	Word 5	m	Index to Final Header ($1 \leq m \leq n$)
	Word 6	n	Number of Headers
	Word 7-1+1	Header(s)	
	Word 1 + 2 and on for items 2 through k		
3 Task Directive	Word 1		Directive
		2	= Suspend Task
		3	= Resume Task
		4	= Abort Task

10-A. DOS-1 to DOS-0 Messages (Continued)

<u>Message Type</u>	<u>Format of the Message Body</u>	
4 Memory I/C	(See M. J. Young Memo MJY-04A)	
5 Modify Memory	Word 1	n Number of Modifications
	Word 2	Address for Modification 1
	Word 3	New value for Address in Word 2
	Word 4	Address for Modification 2
	Word 5	New value for Address in Word 4

	Word 2	Address for Modification n
	Word 2n+1	New value for Address in Word 2n
6 Update Virtual Address Modification Control Word	Word 1	New Virtual Address Modification Control Word
	Bit 15	{ Set: Modification Permitted Clear: Modification Prohibited
7 Update Recording Control Words	Word 1	Recording On Flags
	Word 2	Recording Off Flags

10-B. DOS-0 to DOS-1 Messages

All have format of:

Word 0 - Message Type

Word 1 - n - Message Body

Message Type

Body Format

1 Error Messages	Word 1	Fault Number	} Message Code = 9
	Word 2 - n	Subsidiary Data	
2 Task Directives	Word 1	Action	} MC = 3
	{ 0	Task Loaded as Requested	
	{ 8	Schedule Task	
	{ 2	Suspend Task	
	{ 3	Resume Task	
	{ 4	Abort Task	
	{ 5	Swap Tasks	
	Word 2	Task ID(s) for 8, 2, 3, 4, 5*	
	Word 3	Start Address** (1)	
	Word 4	Start Address (2)	

*Swap Task - Virtual Address 1 in bits 15 - 8, Word 2
Virtual Address 2 in bits 7 - 0, Word 2

Start Address for Virtual Address 1 in Word 2

Start Address for Virtual Address 2 in Word 4

**Schedule Task - Start Address - If ϕ , DOS- ϕ uses address in prologue, Word 2.

NB uses 73 as an intermediate VA during swap.

10-C. DOS-0 to Trace Device (Page 1 of Common Memory, '40)

<u>Word Number</u>	<u>Contents</u>
0	Page Sequence Number
1	Test Number = 2 for Trace Message
2	Subtest Number = $\begin{cases} 0 & \text{Register Dump} \\ 1 & \text{Registers and Memory} \end{cases}$
3	Word Count (n) for memory dump option, must be less than 222 words.
4	Starting address for memory dump
5 - 9	Spare words
10 - 18	Register contents: M = Instruction, P, S, B, X, A, E, I, W
19 - (n + 17)	Memory dump

11. I/O STATUS WORD FORMAT

The I/O status word is divided into four fields: receive word count, receive status, transmit status, last transmit bus.

15	8	7	5	4	1	0
Receive word count		Receive status		Transmit status		Bus

Bits

Status Field Definition

15 - 8	Receive word count for last block received. Does not include the initial header word.	
7 - 5	Receive status	Octal
	100 - Idle	200
	110 - Parity Error on Bus A	300
	101 - Parity Error on Bus B	240
	010 - Incomplete Block A	100
	001 - Incomplete Block B	040
	000 - Receive Buffer Full	000
	All others - Illegal	
4 - 1	Transmit Status	Octal
	1111 - Transmit Triggered	036
	1101 - Bus Busy	032
	1011 - Arbitration Fault	026
	1001 - Reply Fault	022
	0111 - Receiver Busy	016
	0101 - Parity Error	012
	0011 - Timing Fault	006
	0001 - Done	002
	0000 - Idle	000
	All others - Illegal	
0	Last Transmit Bus	
	0 - Bus A	
	1 - Bus B	

12. FAULT MESSAGES/FAULT BITS TO DOS-1

<u>Imp.</u> *	<u>No.</u>	<u>Fault</u>	<u>Fault Data</u>	<u>Routine</u>
X	1	Illegal Instruction	Address, Instru	DOILLI
X	2	Illegal Address	Address,	DOILLA
X	3	Stack Overflow	Overflow Indicator	DOSTAK
X	4	Invalid System Call	Call #, Packet Addr.	DOSYSC
	5	System Call I/O Error	P. A. , Status	DOSYSC
X	6	Input Error	Status Word, Header	DOIN
	7	Unable to Queue Input		
	8	SPARE		
	9	Unable to Queue Input		DOLOAD
	10	I/O Error or Checksum Error		DOLOAD
	11	Unable to Queue Input Request	Packet Address	DOIR
	12	Unable to Queue Output Request	Packet Address	DOIR
	13	Unable to Queue Output Request	Packet Address	DOIOR
X	14	Incorrect Header List	First Header	DOIN
X	15	Invalid Message Code	Code, Header	DOIN
	16	Task Abort		DOPIM
X	17	I/O Trace Error Status Word	Status Word	DOTRAC
X	18	Undefined Execute Message	Message Number	DOPIM
X	19	Unwanted Unsolicited Input	Header Word	DOPUNS
X	20	Execute Message not from Exec.	Header	DOPIM
X	21	Returned Message	Header	DOPIM
X	28	I/O Error in Data Recording	N/A	DOSYS 7
X	29	Buffer Overflow in Data Record.	N/A	DOSYS 7
X	30	Faulty Extraction Point Number	N/A	DOSYS 7
X	31	DOS-Ø PROM Checksum Error	N/A	DOCLCK

*Currently Implemented

12. FAULT MESSAGES/FAULT BITS TO DOS-1

<u>Imp.</u> *	<u>No.</u>	<u>Fault</u>	<u>Fault Data</u>	<u>Routine</u>
X	1	Illegal Instruction	Address, Instru	DOILLI
X	2	Illegal Address	Address,	DOILLA
X	3	Stack Overflow	Overflow Indicator	DOSTAK
X	4	Invalid System Call	Call #, Packet Addr.	DOSYSC
	5	System Call I/O Error	P. A. , Status	DOSYSC
X	6	Input Error	Status Word, Header	DOIN
	7	Unable to Queue Input		
	8	SPARE		
	9	Unable to Queue Input		DOLOAD
	10	I/O Error or Checksum Error		DOLOAD
	11	Unable to Queue Input Request	Packet Address	DOIIR
	12	Unable to Queue Output Request	Packet Address	DOIIR
	13	Unable to Queue Output Request	Packet Address	DOIOR
X	14	Incorrect Header List	First Header	DOIN
X	15	Invalid Message Code	Code, Header	DOIN
	16	Task Abort		DOPIM
X	17	I/O Trace Error Status Word	Status Word	DOTRAC
X	18	Undefined Execute Message	Message Number	DOPIM
X	19	Unwanted Unsolicited Input	Header Word	DOPUNS
X	20	Execute Message not from Exec.	Header	DOPIM
X	21	Returned Message	Header	DOPIM
X	28	I/O Error in Data Recording	N/A	DOSYS 7
X	29	Buffer Overflow in Data Record.	N/A	DOSYS 7
X	30	Faulty Extraction Point Number	N/A	DOSYS 7
X	31	DOS-Ø PROM Checksum Error	N/A	DOCLCK

* Currently Implemented

13. COMMON ELEMENT PROGRAM STATUS WORD (PSW)

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
 * * * * * Unused * * * * *

RS	OF	T	T	M	E
----	----	---	---	---	---

<u>Bits</u>	<u>Field Name</u>	<u>Description</u>	<u>Values</u>
7 - 5	RS	Register Set Number	0 - 7
4	OF	Overflow Indicator	0 - No overflow 1 - Overflow
2	T	Trace Indicator	0 - No trace 1 - Trace
1	M	Program Mode	0 - User (non-private) 1 - System (private)
0	E	Interrupt Enable*	0 - Disabled 1 - Enabled
3	I	Instruction Set No.	0 - Set 0 1 - Set 1

*Interrupts include:

- i. Output complete
- ii. Input pending
- iii. Clock interrupt
- iv. Trace interrupt (?)

14. SYSTEM STATUS RETURN MESSAGE FORMAT (CE/IOC/CM)

<u>Word Number</u>	<u>Format/Contents</u>
0	Bits 15 - 9 Status of Card Bit 8 Error Summary Bit Bits 7 - 6 Card Type 00 Illegal 01 CE 10 IOC 11 CM Bits 5 - 0 Virtual Address
1 - 2 - 3	Error Indicators - 1 bit per error

NOTE: Error summary bit in word 0 is set whenever one or more error indicator bits are set in words 1 through 3.

See Memo H. E. T. Connell HETC-04 for more details on status return messages.

Release of DOS-0 Version 22

Version 22 of DOS- ϕ was released on March 15, 1979 into all CE's in the Sudbury Test Facility. The only major update item for this version is that two new system requests have been added to the list of services provided by DOS- ϕ .

These new system requests are:

System Request 10 - Dequeue an output request

System Request 11 - Dequeue an input request

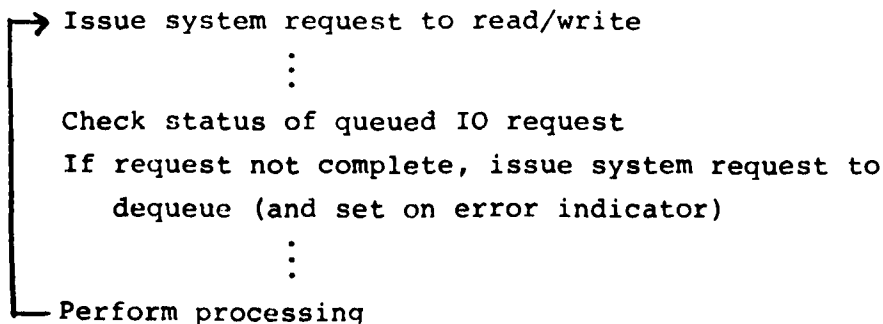
Both of these requests require a packet address in the A register at the time the request is issued. Upon return from the request, bits 13 and 14 of the status word of the packet (word ϕ) may be interrogated in order to obtain some information about the dequeuing request. In particular,

Bit 13 set implies that the specified packet address was found in the queue and that the request was properly dequeued

Bit 14 set indicates that the packet address was not found in the queue and that no action was taken

These system requests have been implemented to alleviate the problem of IO request entries (in DOS- ϕ) becoming exhausted in the case where a user "time's-out" on an IO request and reinserts another request into the queue.

Thus, the following sequence should be used for maintaining maximum IO entries in DOS- ϕ :



Any problems relating to these new system requests should be reported to me.

B-2

IDOS-0 Hierarchy and Subroutine Usage

AD-A108 253

RAYTHEON CO WAYLAND MA EQUIPMENT DIV

F/G 17/9

RAD EQUIPMENT INFORMATION REPORT. FAULT TOLERANT WEATHER RADAR --ETC(II)

MAR 81 M J YOUNG, A J JASODNIK

F1962A-75-C-0113

UNCLASSIFIED

ER81-4053

AFGL-TR-81-0086

NL

4 x 4

1/2 inch



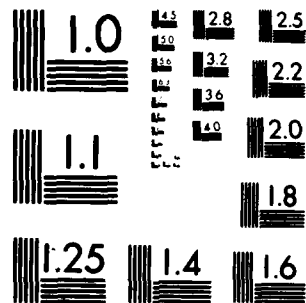
END

DAYS

FILED

1/2

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A.

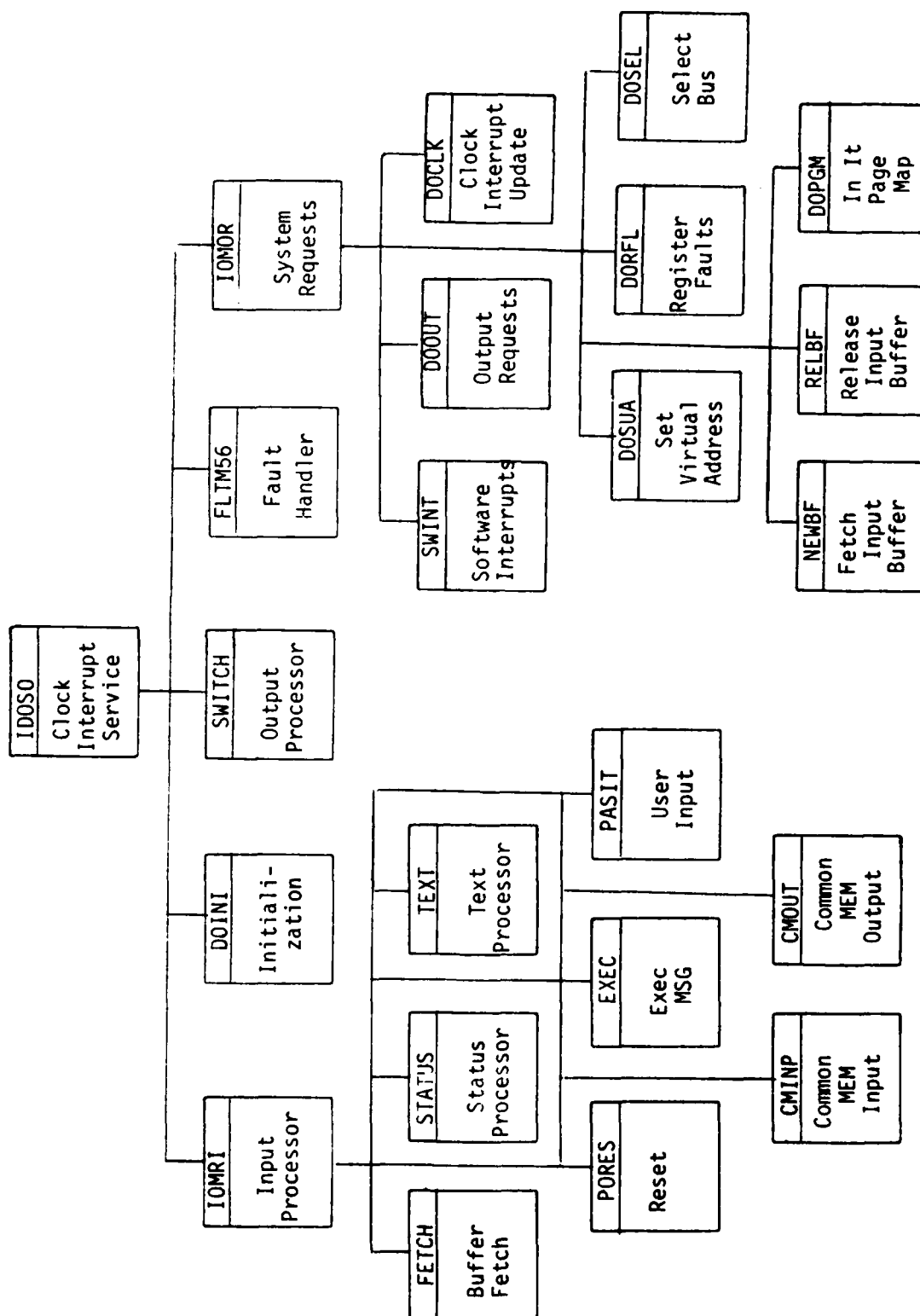


Figure B-2. IDOS-0 Hierarchy

Table B-2. IDOS-0 Subroutines and Functions

<u>Name</u>	<u>Function</u>	<u>Parameters</u>	<u>Returned Values</u>
DOINI	IDOS-0 Initialization starts Interval timer interrupts, sets up tables, calls IUSER		
DOIDL	IDOS-0 Idle Loop Entered From Fatal Errors		
IDOSO	Interval Timer Service Routine - Polls TIE, Updates User Clock		
IOMRI	Input Message Processor		
PORES	Reset Function Handler		
PASIT	Passes Buffer to User		
FMSG	Outputs Fault Message to DOS-1		
SRTND	Status Return Processor		
TEXT	Text Message Processor		
LOPROT	List to Protected Screen	HL = Address of character D = Y Boundary E = X Boundary	None
LVA	Load Virtual Address Handler		
RTND	Returned Block Processor		
EXEC	Executive Message Handler		
FETCH	Buffer Fetch - Constructs Buffer Packet	HL = Buffer Packet Pointer	

Table B-2. IDOS-0 Subroutines and Functions (Cont'd)

<u>Name</u>	<u>Function</u>	<u>Parameters</u>	<u>Returned Values</u>
ICC	Checks for Inter-Cluster Communication (multiple Headers)	HL = Header List	Carry = Yes No Carry = No
CMINP	Common Memory input Handler		
CMOUT	Common Memory Output		
SWTHD	Updates Headers to Return a Message	DE = Address to which Headers will go	
SWITCH	Switches Source & Destination in a Header	BC = Header	
STATUS	Status request Processor		
DECODE	Determines if CM Page is on Disk or in RAM sets up FCS Command and Buffer	BC = Page No	DE - Buffer ADDR
IOMOR	System Service Handler	A = Call Number BC } DE } Service Parameters HL }	
DOOUT	Message Output Handler	B = Header Count C = Word Count DE = Header List ADOR HL = Data Buffer ADOR	A = Status
DOCLK	User Clock Update	BC = New Options DE = New Period	
DOSUA	Set Virtual ADOR	B = New VA	
DOSEL	Select Bus	B = New Bus Option	
DORFL	Register Fault	B = Fault Number	

Table B-2. IDOS-0 Subroutines and Functions (Cont's)

<u>Name</u>	<u>Function</u>	<u>Parameters</u>	<u>Returned Values</u>
NEWBF	Fetch New Input Buffer Packet	HL = Address to Store Packet	A = Status of Packet
RELBF	Release Packet		
DOPGM	Initialize Page Map		
SWINT	Software Interrupt Handler		
FLTMSG	Output Fault Message	A = Fault Number	
SCALL	Invalid System Call	E = Sys Call No.	
ISERR	I/O Status Error Handler		
IVAMC	Invalid Message Code Handler		
TABRT	Task Abort Handler		
UDIDO	Invalid Exec Message	E = MSG Type	
EXNDI	Exec MSG Not From DOS-1 Handler		
DOIER	Input Message Error - Hard Failure		Drop to Idle Loop
DOIOR	Output Error - Hard Failure		Drop To Idle

IDOS-0 Major Variables and Data Structures

<u>Variable Name</u>	<u>Size Bytes</u>	<u>Starting Address</u>	<u>Description</u>
CONTBL	512	F400	Configuration Table, Not Used At Present
PGMAP	512	F600	Page Map
PGBUF	512	F800	Temporary Page Buffer (Available to User)
PG000	512	FA00	Page 0 Buffer
PG001	512	FC00	Page 1 Buffer
PG002	512	FE00	Page 2 Buffer
STAT0	2	E008	} IDOS-0 Task Status Words, Sent to DOS-1 Upon Status Request Message
STAT1	2	E00A	
STAT2	2	E00C	
STAT3	2	E00E	
NCLK	2	E006	User Clock, Current Value
IBUFP	2	E010	IDOS-0 Input Packet Pointer
CBPTR	2	E012	Current Buffer Pointer
CHDRCT	1	E014	Current Header Count
CHADR	2	E015	Current Header Address
CSRC	1	E017	Current Source
CSRCPT	2	E018	Current Source Address
CWDCT	1	E01A	Current Word Count
MSGCD	1	E01B	Current Message Code
CCMPG	1	E01C	Current CM Page Number
BUFOF	1	E01D	Packet 0 Buffer Full Flag
PTRO	2	E01E	Packet 0 Buffer Pointer
HDCTO	1	E020	Packet 0 Header Address

IDOS-0 Major Variables and Data Structures (Con't)

<u>Variable</u>	<u>Size</u>	<u>Starting</u>	<u>Description</u>
HDADO	2	E021	Packet 0 Header Address
SRCO	1	E023	Packet 0 Source
SRCPO	2	E024	Packet 0 Source Address
WDCTO	1	E426	Packet Word Count
MSGO	1	E027	Packet Message Code
CMPGO	1	E428	Packet 0 CM Page No.
BUF1F	12	E029	Packet 1
BUF2F	12	E035	Packet 2
TEMPO	1	E042	} Temporary Storage
TEMP1	1	E043	
TEMP2	1	E044	
SRETF	1	E045	Status Return in Progress Flag
SBUSF	1	E046	XMIT Bus Option
ALTBF	1	E047	Output Failure Retry in Progress
RETRF	1	E048	Retry Authorization on Output
SVCFG	1	E049	Software Interrupt in Progress
UBUFP	2	E04A	User Input Packet Pointer
RESET	1	E04C	Reset in Progress
OSTK	2	E04D	User Stack Pointer Save Area
IHDRO	16	E04F	Header Buffer - Packet 0

IDOS-0 Major Variables and Data Structures (Con't)

<u>Variable</u>	<u>Size</u>	<u>Starting</u>	<u>Description</u>
IHDR1	16	E24F	Header Buffer - Packet 1
IHDR2	16	E44F	Header Buffer - Packet 2
IBUF0	496	E05F	Data Buffer - Packet 0
IBUF1	496	E25F	Data Buffer - Packet 1
IBUF2	496	E45F	Data Buffer - Packet 2
BUFAD	480	E661	Output Buffer for CM Input Handler

B-3

IDOS-1 Subroutines

Table B-3. IDOS-1 Subroutines

<u>Name</u>	<u>Description</u>
IDOS1	Main entry point--performs initialization
DIENT	Pulse-Pair processing entry point
RESTART	Restart entry point (after RESET)
SYSIN	Entry point for processing CONT command
STARTUP	Loads all tasks in preparation for system startup
CLKIN	Init's all clocks (system clock, spare rotation clock, etc)
ERSCR	Erases the screen, then enters the command processor
CMDPRO	Command processor
CMDER	Illegal command entry point
PARER	Illegal Parameter entry point HL = position of error
NOPAR	Missing Parameter entry point HL = position of error
ERPOS	Outputs position of error in command line HL = position of error
CLKSVC	Clock interrupt service routine
RECNF	Reconfiguration routine A = VA to be reset
STPOLL	Status poll routine entry point
CLEAR	Erases screen, then repaints status on display
LSTAT	Paints status on display (without erasing first)
MSGP?	Message input interrupt handler
MSCD3	Message Code 3 handler
STPRO	Status return (message code 8) handler
DSPMSG	Displays message to screen
FAULT	Fault message (message code 9) handler
RLBUFF	Release input message buffer to IDOS-0
STNUP	Updates STNDX table A = old VA B = new VA

Table B-3. IDOS-1 Subroutines (con't)

<u>Name</u>	<u>Description</u>
	0 = card type
	1 = VA
	2 = Status byte 1
	3 = Status byte 2
	4 = job status
	5 = task ID
	6 = task VA
	7 = status poll count
	C = entry number (SA)
	DE,HL modified
SYSLD	Fetch SYSTBL entry (value returned in A)
	B = word in entry
	C = entry number (SA)
	DE,HL modified
FSTNX	Fetch STNDX entry (returned in A)
	A = VA
	DE,HL modified
BREAK	Checks for keyboard (BREAK) key depressed
	Z = break is detected
RILST	Remove VA from idle list
	(SP+2) = VA
TASK	A,B,DE,HL modified
	CARRY = error in task load
CESRT	Starts all loaded tasks
	CARRY = error in task start
CESPR	Checks spare list for loadable CEs
	CARRY = none found
TSKER	Task load error (fatal)

Table B-3. IDOS-1 Subroutines (con't)

<u>Name</u>	<u>Description</u>
IOCER	IOC failure (nonfatal)
YSER	System error (fatal)
TSKLD	Loads a task DE = task ID TSRL = task VA
SUSTK	Suspend all tasks
RESTD	Resume all tasks
ABRTT	Abort all tasks
ABRT	Abort a task A = VA to be aborted C = SA of card to be aborted
IOCST	IOC startup routine
SNPAR	Send signal processing parameters to CE A = VA
STPSP	Stop signal processing in CEs A = VA CARRY = not stopped due to error
SPROT	Performs spare rotation
PIUPD	Page 1 Update interrupt handler (displays CE memory dump or trace data)

The following routines perform the command processor functions

RES	Performs RES command
DIR	Performs DIR command
CLR	Performs CLR command
SAV	Performs SAV command
LOA	Performs LOA command
PRNT	Perform PRINT command
RADIX	Performs RADIX command

Table B-3. IDOS-1 Subroutines (con't)

<u>Name</u>	<u>Description</u>
CYB	Performs CYB command
INIT	Performs INI command
TSK	Performs TSK command
ST	Performs ST command
DSP	Performs DSP command
POMOD	Sets up Page 0 for DSPM and MODM commands
MODFY	Performs MOD command
MSG	Performs MSG command
SVA	Performs VA command
SBUSS	Performs BUS command
CMR	Performs CMR command
CMW	Performs CMW command
CMREQ	Common processing for CMR and CMW
SCH	Performs SCH command
CON	Performs CON command
TRA	Performs TRA command
TSP	Performs TSP command
TRS	Performs TRS command
SETT	Performs SET command
STQ	Performs STQ command
CTQ	Performs CTQ command
ABOTR	Performs ABORT command
CPP	Performs CONT command

Table B-4. FTWRP Subroutines and Functions

<u>Name</u>	<u>Description</u>
START	Initialization entry point
DWELL	Main processing loop and dwell-level entry point
FORMAT	Formats and outputs data to OS
OUTDAT	Outputs a message to OS
OSLINT	Outputs SLINT data to next CE
IQPULS	Coherent channel pulse-level processor
LZPULS	Reflectivity channel pulse-level processor
DOS1DR	DOS-1 directives handler
CHGPAR	Changes signal processing parameters
CHGVAD	Sets up CE to change VA for spare rotation
CHGLT1	Loads Range Normalization look-up table
CHGLT2	Loads Tangential Shear look-up table
SUSPEN	Suspends signal processing
RESTRT	Restarts signal processing
CHGLUT	Common processing for CHGLT1 and CHGLT2
SLNTIN	Inputs new SLINT data from previous CE
TSKINI	Initializes all packet addresses and constants

B-4

IOC Continuous Input Mode Subroutines

IOC Continuous Input Mode Subroutines

<u>NAME</u>	<u>DESCRIPTION</u>
RECVIN	Determines the wordcount of the next block of data and places it in R6
RECBK	Used in "ping-ponging" X and Y RAMS by setting one RAM in input mode, while preparing other for output.
PRETRN	Assumes wordcount is in R6. Writes the header word and wordcount in RAM in preparation for transmission to a CE.
TRANSM	Starts transmission in one RAM, then prepares other RAM for receiving from Input Synchronizer. Waits until transmission is complete, and checks transmitter status.
SWRAM	Used in copying virtual address and wordcount list from original RAM to other RAM.
INPSEL	Determines whether coherent channel or reflectivity channel data is to be input next, and places value of R1 or R5 into R2 accordingly. R2 will thus have the address of the next virtual address and wordcount to be used.
ADDRAM	Addresses the proper RAM at the address contained in register R2.
CMTN	Sets up receiver to begin receiving at the third word in the current RAM (leaves two words for header and word count). Starts receiver, then the transmitter is started.

APPENDIX C

"Notes on Circular Vectoring"

(D. A. Syiek memo #DAS-02)



FORM 10-0557 (2-65) BOND

DIVISION EQUIPMENT

Operation EDL

Department ADL - Advanced Electronic Techniques Distribution cc

To A. J. Jagodnik, Jr.

From D. A. Syiek

Subject Notes on Circular Vectoring

Classification Unclassified

Contract No. F19628-78-C-0113

File No. EM78-0422

Memo No. DAS-02

Date 2 August 1978

ABSTRACT

One of the concluding steps in the algorithm implemented by the FTWRP being developed for AFGL to replace existing hardware, is the calculation of $\tan^{-1}(y/x)$. In the present system, special-purpose hardware utilizes a ROM look-up table to find this number. However, with the proposed new system, such an approach represents an unnecessary cost increase, as the processor itself is capable of being programmed for this calculation. This paper is a short treatise on the method used to implement this operation.

Initially, traditional Cordic was explored as a possible solution. This method visualizes x and y as coordinates of a vector, the angle of which is equal to $\tan^{-1}(y/x)$. If one rotates this vector about the origin until its y component is zero and its x is max. positive, then the original angle of the vector is found by summing the amount of rotation required to reach this state. The angular rotation is performed by rotating towards the positive x axis a little at a time, using fixed angles of decreasing magnitude. Any series of angular rotations is acceptable provided the sum of the remaining series at any point is at least as large as the amount yet to be rotated. This condition necessitates that the sum of the entire series be $\geq 180^\circ$ in order to accommodate the entire range of -180° to $+180^\circ$. This also means that the sum of the remaining series at any point must be at least as great as the current angle of rotation. In more formal terms:

Theorem:

Given a vector of angle θ , in the range -180° to $+180^\circ$, any series of rotations $\{\pm\alpha_1, \pm\alpha_2, \pm\alpha_3, \dots, \pm\alpha_n\}$ made towards the positive x axis will

cause y to converge to zero if: (1) $\sum_{i=1}^n \alpha_i \geq 180^\circ$ and (2) $\sum_{i=j+1}^n \alpha_i \geq \alpha_j$

for $j = 1, 2, 3, \dots$

"Proof:"

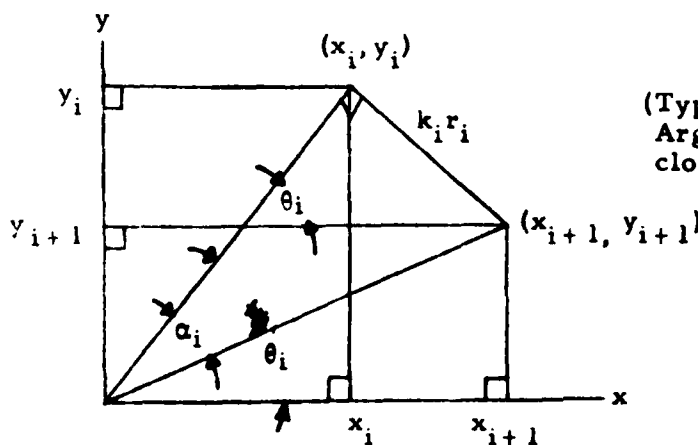
(1) Obvious if allowed to range from -180° to $+180^\circ$. (2) Suppose that at rotation i , $\theta_i \geq 0$, (the proof of the case of $\theta_i < 0$ is identical) and we want to rotate α_i degrees. Then $\theta_{i+1} = \theta_i - \alpha_i$. θ_{i+1} is either ≥ 0 or < 0 depending on the relative magnitudes of θ_i and α_i . If θ_{i+1} is < 0 , then θ_{i+1} can be as negative as $-\alpha_i$ (since θ_i was as small as zero) and the sum of the remaining rotations must be $\geq \alpha_i$ to correct for this error. If θ_{i+1} is ≥ 0 , then the sum of the remaining rotations must be $\geq \theta_{i+1}$ for the same reason. Since the fastest converging series usually rotate at least half the remaining angle at each step, we shall assume that if $\theta_{i+1} \geq 0$ then it is also $\leq \alpha_i$. Thus in both cases, the sum of the remaining rotations must be $\geq \alpha_i$.

One can see that given these requirements, the fastest converging series of angular rotations is the one given by $\alpha_i = 180 \cdot 2^{-i}$ where $i = 1, 2, 3, \dots$

Cordic chooses its series as follows. The first rotation is chosen as 90° and new x, y coordinates are computed directly:

$$\left. \begin{array}{l} x_2 = y_1 \\ y_2 = -x_1 \end{array} \right\} \text{(for clockwise rotation)}$$

Succeeding rotations are chosen based on the following:



(Typical clockwise rotation.
Argument is the same for counter-
clockwise rotation, only signs change.)

$$\begin{aligned} x_i &= r_i \cos \theta_i \\ y_i &= r_i \sin \theta_i \end{aligned}$$

(1)

$$\begin{aligned}x_{i+1} &= x_i + k_i r_i \sin \theta_i \\y_{i+1} &= y_i - k_i r_i \cos \theta_i\end{aligned}\tag{2}$$

Substituting equation (1) into (2),

$$\begin{aligned}x_{i+1} &= x_i + k_i y_i \\y_{i+1} &= y_i - k_i x_i\end{aligned}\tag{3}$$

$$\tan \alpha_i = \frac{k_i r_i}{r_i} = k_i\tag{4}$$

Cordic chooses k_i such that all math can be done by shifts and adds:
 $k_i = 2^{-(i-1)}$, $i = 2, 3, \dots$. So $\tan \alpha_i = 2^{-(i-1)}$ and $\alpha_i = \tan^{-1} (2^{-(i-1)})$. The resulting series of angles is $\{90^\circ, 45^\circ, 26.5^\circ \dots \tan^{-1} (2^{-(i-1)})$ and can be shown to fulfill the two requirements set down in the theorem. When implementing a Cordic routine, these angles must be kept in a table and added (or subtracted, depending on direction of rotation) from the rotational sum at each step of rotation.

Cordic has the disadvantage of being a "growing" algorithm in the sense that the length of the vector increases at each step of rotation. In the limit, the final vector may be as much as 1.65 times as long as the initial one. If this disadvantage is coupled with the fact that a non-growing rotation naturally results in a final x coordinate which may be $\sqrt{2}$ times as great as the largest initial coordinate (case of 45° vector), it means that in order to avoid overflow, x, y inputs, which are allowed to range over the entire precision of the computer, must be scaled down by a factor of $1/(\sqrt{2} \cdot 1.65)$ or .43. With traditional machines this would be done by right shifting arithmetically two places and would mean a loss of two bits of precision.

An algorithm similar to Cordic which converges faster can be discovered by re-examining equations (3) and (4). Choose α_i to be that of the fastest converging series mentioned earlier. Thus $\alpha_i = 180 \cdot 2^{-i}$ and k_i solves to be equal to $\tan(180 \cdot 2^{-i})$. Note now, that this approach requires more than just shifts to do its rotations. The equations for x_{i+1} and y are now:

$$\left. \begin{aligned} x_{i+1} &= x_i + \tan(180 \cdot 2^{-i}) y_i \\ y_{i+1} &= y_i - \tan(180 \cdot 2^{-i}) x_i \end{aligned} \right\} \quad \text{(clockwise rotation)} \quad (5)$$

In the FTWRP, however, such divisions are made easy using the TRW MPY-16AJ multiplier.

When implementing an algorithm of this type, a table of $\tan(180 \cdot 2^{-i})$ must be kept and referenced for each rotation. Angles of rotation may be calculated by shifts since each angle is half its predecessor.

Growth on this algorithm limits to a factor of about $1/(\sqrt{2} \cdot 1.58)$ or .45.

This algorithm has the advantage of fast convergence with only two uses of the multiplier per iteration. Also since only a tan table need be kept, memory usage is at a minimum.

A third algorithm uses the fast convergence and a non-growing rotation. Non-growing rotation is possible through the use of the standard trigonometric rotation equations:

$$\left. \begin{aligned} x_{i+1} &= x_i \cos \alpha_i + y_i \sin \alpha_i \\ y_{i+1} &= y_i \cos \alpha_i - x_i \sin \alpha_i \end{aligned} \right\} \quad \text{(clockwise rotation)} \quad (6)$$

Where α_i is again determined by $\alpha_i = 180 \cdot 2^{-i}$, $i = 1, 2, 3, \dots$

Unclassified
DAS-02
2 August 1978
Page 5

In this case, tables of $\cos \alpha_i$ and $\sin \alpha_i$ must be kept and referenced at each rotation. Again angles of rotation can be found by shifts since each is half its predecessor. Note also, that the TRW multiplier makes possible the divisions on which this algorithm is based.

Scaling problems are limited to the $\sqrt{2}$ factor discussed earlier. Thus, inputs must be scaled by $1/\sqrt{2}$ or .707 before implementing the algorithm. This is a loss of 1 bit of precision if a single right shift arithmetic is done.

Conclusion:

After a careful study of the relative speeds of execution on the FTWRP, it was found that the second algorithm ran the fastest but was closely followed in speed by the third. It was decided to implement the third algorithm due to its increased accuracy.

D. A. Syiek
D. A. Syiek
Advanced Electronic Techniques
Wayland Box M9, x2737

DAS/lld

cc: K. Glover (3) AFGL
K. Banis "
G. Armstrong "

P. C. Barr
G. A. Works
M. J. Young

APPENDIX D

FTWRP System Timing Considerations

There are two major considerations involved in determining the maximum capabilities of the FTWRP system. The first of these is the IOC Continuous Input Mode timing, and the other is the speed of the CE-resident pulse-pair algorithms.

While in the Continuous Input Mode, the IOC must receive and transmit two packets of data for every block of range cells (in the normal system, 5 blocks). Therefore, there must be 10 packet transfers for each radar pulse interval.

The maximum time the select lines (SEL0 and SEL1) may remain inactive (that is, the time when neither is active and switching from one to the other is taking place), is determined by the following formula:

$$T_{sel}(\mu\text{sec}) = \frac{T_p * (N_{rc}/N_{ce}) - 2 * [f * (N_{rc}/N_{ce})]}{2}$$

where N_{ce} is the number of processing CEs (normally 5)

N_{rc} is the number of range cells processed per pulse repetition interval

T_p is the time per range cell (in $\mu\text{seconds}$)

and f is the FTSP bus data bandwidth in $\mu\text{sec/word}$.

Table D-1 enumerates various values for T_{sel} with all possible values for N_{rc} , T_p , and bus bandwidth. Empirical analysis shows that the actual inactive select time in FTWRP is approximately 32-38 μsec . The bus bandwidth is somewhat greater than 4 Mwords/sec, but less than 5 Mwords/sec (the actual value must still be determined). Therefore, according to Table D-1, depending on the actual bus bandwidth, 1024 range cells @ 1 μsec is certainly possible, and 768 range cells @ 1 μsec is probable. Other combinations, such as $N_{rc} = 256$ @ 1 μsec , are clearly unrealizable.

The CE timing is affected by four major components of processing: 1) the reflectivity pulse-level processing, 2) the coherent pulse-level processing, 3) the dwell-level processing, and 4) the output formatting operation. These times are determined by the following formulae (all values are in number of microcycles):

Reflectivity:

$$T_r = [(N_{si} - 1) (129 + 3N_{rc})] + (247 + 3N_{rc}) + 100N_{si}$$

Coherent:

$$T_c = [(N_{si} - 1) (139 + 11N_{rc})] + (245 + 11N_{rc}) + 100N_{si}$$

Dwell Processing:

$$T_d = 614 + 111N_{rc}$$

Format and Output:

$$T_f = 1521 + 9693(N_{rc}/79) + 106(N_{rc} \bmod 79)$$

where:

N_{si} = number of pulses per dwell

N_{rc} = number of range cells per CE

/ = integer divide (truncated result)

The total time required to perform on a dwell of information is thus

$$T = T_{\mu c}(T_r + T_c + T_d + T_f)$$

where $T_{\mu c}$ is the CE microcycle time (for ease of computation, 250 nsec is used here, but actually it is closer to 240-245 nsec).

Using the above formulas, assuming $N_{rc} = 768$, the worst-case maximum PRF of the radar transmitter is determined to be:

$$\begin{aligned} \text{PRF} &= 1236 \text{ Hz} & (N_{si} = 64) \\ &1358 \text{ Hz} & (N_{si} = 128) \end{aligned}$$

Table D-1. Maximum Allowable Inactive Select Times

Nrc	Tp	Tsel	
		f = .25	f = .30
256	1	12.5	9.9
512	1	25.0	19.8
768	1	37.5	29.7
1024	1	50.0	39.6
256	2	38.0	35.4
512	2	76.0	70.8
768	2	114.0	106.2
1024	2	152.0	141.6

APPENDIX E

Look-Up Tables

Listing E-1. Range Normalization Table Generator

```
10 B=120:C=100
20 P1=1/(LOG(10)):P2=6/(P1*LOG(2)*0.4)
30 FOR C=100 TO 101
40 A=53248:M=0
45 A=A-65536
50 D=15
60 IF C=101 THEN D=31
70 POKE A,0:POKE A+1,5
80 A=A+2
90 FOR L=1 TO 5
100 Z=49152-65536
101 POKE Z,0:POKE Z+1,0:Z=Z+2
110 FOR N=MT0 M+239
111 IF N=0 THEN 150
112 IF N>=1024 THEN 155
120 X=INT(P1*LOG(N)*P2)+D
130 POKE Z,0:POKE Z+1,X
131 PRINTN,X
140 Z=Z+2
150 NEXT N
155 E$="SAV1:PAGE."
160 PLOT27:PLOT4
170 PRINT E$+RIGHT$(STR$(B),3)+" C000 1E0"
180 PLOT27:PLOT27
190 B=B+1
200 M=M+240
210 POKE A,0:POKE A+1,B-1
220 POKE A+2,0:POKE A+3,240
230 A=A+4
240 NEXT L
250 POKE A-1,64
260 PLOT27:PLOT4
270 PRINT E$+RIGHT$(STR$(C),3)+" D000 1E0"
280 PLOT27:PLOT27
290 NEXT C
300 END
```

Table E-1. Range Normalization Tables ($T_p = 1\mu S$)

CELL #	CONTENTS	CELL #	CONTENTS	CELL #	CONTENTS
1	15	53	100	105	115
2	29	54	101	106	115
3	38	55	101	107	116
4	44	56	102	108	116
5	49	57	102	109	116
6	53	58	102	110	116
7	57	59	103	111	116
8	60	60	103	112	117
9	62	61	103	113	117
10	64	62	104	114	117
11	66	63	104	115	117
12	68	64	105	116	117
13	70	65	105	117	118
14	72	66	105	118	118
15	73	67	105	119	118
16	74	68	106	120	118
17	76	69	106	121	118
18	77	70	106	122	118
19	78	71	107	123	119
20	79	72	107	124	119
21	80	73	107	125	119
22	81	74	108	126	119
23	82	75	108	127	119
24	83	76	108	128	120
25	84	77	109	129	120
26	85	78	109	130	120
27	86	79	109	131	120
28	87	80	109	132	120
29	87	81	110	133	120
30	88	82	110	134	120
31	89	83	110	135	121
32	90	84	110	136	121
33	90	85	111	137	121
34	91	86	111	138	121
35	91	87	111	139	121
36	92	88	111	140	121
37	93	89	112	141	122
38	93	90	112	142	122
39	94	91	112	143	122
40	94	92	112	144	122
41	95	93	113	145	122
42	95	94	113	146	122
43	96	95	113	147	122
44	96	96	113	148	123
45	97	97	113	149	123
46	97	98	114	150	123
47	98	99	114	151	123
48	98	100	114	152	123
49	99	101	114	153	123
50	99	102	115	154	124
51	100	103	115	155	124
52	100	104	115	156	124

Table E-1. Range Normalization Tables ($T_p = 1\mu S$)

CELL #	CONTENTS	CELL #	CONTENTS	CELL #	CONTENTS
157	124	187	128	217	131
158	124	188	128	218	131
159	124	189	128	219	131
160	124	190	128	220	131
161	124	191	128	221	131
162	125	192	128	222	131
163	125	193	128	223	132
164	125	194	128	224	132
165	125	195	129	225	132
166	125	196	129	226	132
167	125	197	129	227	132
168	125	198	129	228	132
169	126	199	129	229	132
170	126	200	129	230	132
171	126	201	129	231	132
172	126	202	129	232	132
173	126	203	129	233	132
174	126	204	130	234	133
175	126	205	130	235	133
176	126	206	130	236	133
177	127	207	130	237	133
178	127	208	130	238	133
179	127	209	130	239	133
180	127	210	130		
181	127	211	130		
182	127	212	130		
183	127	213	131		
184	127	214	131		
185	127	215	131		
186	128	216	131		

Table E-1. Range Normalization Tables ($T_p = 1\mu S$)

CELL #	CONTENTS	CELL #	CONTENTS	CELL #	CONTENTS
240	133	290	137	340	141
241	133	291	137	341	141
242	133	292	137	342	141
243	133	293	137	343	141
244	133	294	137	344	141
245	134	295	138	345	141
246	134	296	138	346	141
247	134	297	138	347	141
248	134	298	138	348	141
249	134	299	138	349	141
250	134	300	138	350	141
251	134	301	138	351	141
252	134	302	138	352	141
253	134	303	138	353	141
254	134	304	138	354	142
255	134	305	138	355	142
256	134	306	138	356	142
257	135	307	138	357	142
258	135	308	139	358	142
259	135	309	139	359	142
260	135	310	139	360	142
261	135	311	139	361	142
262	135	312	139	362	142
263	135	313	139	363	142
264	135	314	139	364	142
265	135	315	139	365	142
266	135	316	139	366	142
267	135	317	139	367	142
268	135	318	139	368	142
269	136	319	139	369	142
270	136	320	139	370	142
271	136	321	139	371	143
272	136	322	139	372	143
273	136	323	140	373	143
274	136	324	140	374	143
275	136	325	140	375	143
276	136	326	140	376	143
277	136	327	140	377	143
278	136	328	140	378	143
279	136	329	140	379	143
280	136	330	140	380	143
281	137	331	140	381	143
282	137	332	140	382	143
283	137	333	140	383	143
284	137	334	140	384	143
285	137	335	140	385	143
286	137	336	140	386	143
287	137	337	140	387	143
288	137	338	141	388	143
289	137	339	141	389	144

Table E-1. Range Normalization Tables ($T_p = 1\mu S$)

CELL #	CONTENTS	CELL #	CONTENTS
390	144	440	146
391	144	441	146
392	144	442	146
393	144	443	146
394	144	444	146
395	144	445	146
396	144	446	147
397	144	447	147
398	144	448	147
399	144	449	147
400	144	450	147
401	144	451	147
402	144	452	147
403	144	453	147
404	144	454	147
405	144	455	147
406	144	456	147
407	145	457	147
408	145	458	147
409	145	459	147
410	145	460	147
411	145	461	147
412	145	462	147
413	145	463	147
414	145	464	147
415	145	465	147
416	145	466	147
417	145	467	148
418	145	468	148
419	145	469	148
420	145	470	148
421	145	471	148
422	145	472	148
423	145	473	148
424	145	474	148
425	145	475	148
426	146	476	148
427	146	477	148
428	146	478	148
429	146	479	148
430	146		
431	146		
432	146		
433	146		
434	146		
435	146		
436	146		
437	146		
438	146		
439	146		

Table E-1. Range Normalization (con't)

CELL #	CONTENTS	CELL #	CONTENTS	CELL #	CONTENTS
480	148	530	150	580	152
481	148	531	150	581	152
482	148	532	150	582	152
483	148	533	150	583	152
484	148	534	150	584	152
485	148	535	150	585	152
486	148	536	150	586	152
487	148	537	151	587	152
488	148	538	151	588	152
489	149	539	151	589	153
490	149	540	151	590	153
491	149	541	151	591	153
492	149	542	151	592	153
493	149	543	151	593	153
494	149	544	151	594	153
495	149	545	151	595	153
496	149	546	151	596	153
497	149	547	151	597	153
498	149	548	151	598	153
499	149	549	151	599	153
500	149	550	151	600	153
501	149	551	151	601	153
502	149	552	151	602	153
503	149	553	151	603	153
504	149	554	151	604	153
505	149	555	151	605	153
506	149	556	151	606	153
507	149	557	151	607	153
508	149	558	151	608	153
509	149	559	151	609	153
510	149	560	151	610	153
511	149	561	151	611	153
512	150	562	152	612	153
513	150	563	152	613	153
514	150	564	152	614	153
515	150	565	152	615	153
516	150	566	152	616	154
517	150	567	152	617	154
518	150	568	152	618	154
519	150	569	152	619	154
520	150	570	152	620	154
521	150	571	152	621	154
522	150	572	152	622	154
523	150	573	152	623	154
524	150	574	152	624	154
525	150	575	152	625	154
526	150	576	152	626	154
527	150	577	152	627	154
528	150	578	152	628	154
529	150	579	152	629	154

Table E-1. Range Normalization (con't)

CELL #	CONTENTS	CELL #	CONTENTS
630	154	680	156
631	154	681	156
632	154	682	156
633	154	683	156
634	154	684	156
635	154	685	156
636	154	686	156
637	154	687	156
638	154	688	156
639	154	689	156
640	154	690	156
641	154	691	156
642	154	692	156
643	154	693	156
644	154	694	156
645	154	695	156
646	155	696	156
647	155	697	156
648	155	698	156
649	155	699	156
650	155	700	156
651	155	701	156
652	155	702	156
653	155	703	156
654	155	704	156
655	155	705	156
656	155	706	156
657	155	707	156
658	155	708	157
659	155	709	157
660	155	710	157
661	155	711	157
662	155	712	157
663	155	713	157
664	155	714	157
665	155	715	157
666	155	716	157
667	155	717	157
668	155	718	157
669	155	719	157
670	155		
671	155		
672	155		
673	155		
674	155		
675	155		
676	156		
677	156		
678	156		
679	156		

Table E-1. Range Normalization (con't)

CELL #	CONTENTS	CELL #	CONTENTS	CELL #	CONTENTS
720	157	770	158	820	160
721	157	771	158	821	160
722	157	772	158	822	160
723	157	773	158	823	160
724	157	774	158	824	160
725	157	775	158	825	160
726	157	776	158	826	160
727	157	777	159	827	160
728	157	778	159	828	160
729	157	779	159	829	160
730	157	780	159	830	160
731	157	781	159	831	160
732	157	782	159	832	160
733	157	783	159	833	160
734	157	784	159	834	160
735	157	785	159	835	160
736	157	786	159	836	160
737	157	787	159	837	160
738	157	788	159	838	160
739	157	789	159	839	160
740	157	790	159	840	160
741	157	791	159	841	160
742	158	792	159	842	160
743	158	793	159	843	160
744	158	794	159	844	160
745	158	795	159	845	160
746	158	796	159	846	160
747	158	797	159	847	160
748	158	798	159	848	160
749	158	799	159	849	160
750	158	800	159	850	160
751	158	801	159	851	160
752	158	802	159	852	161
753	158	803	159	853	161
754	158	804	159	854	161
755	158	805	159	855	161
756	158	806	159	856	161
757	158	807	159	857	161
758	158	808	159	858	161
759	158	809	159	859	161
760	158	810	159	860	161
761	158	811	159	861	161
762	158	812	159	862	161
763	158	813	160	863	161
764	158	814	160	864	161
765	158	815	160	865	161
766	158	816	160	866	161
767	158	817	160	867	161
768	158	818	160	868	161
769	158	819	160	869	161

Table E-1. Range Normalization (con't)

CELL #	CONTENTS	CELL #	CONTENTS
870	161	921	162
871	161	922	162
872	161	923	162
873	161	924	162
874	161	925	162
875	161	926	162
876	161	927	162
877	161	928	162
878	161	929	162
879	161	930	162
880	161	931	162
881	161	932	162
882	161	933	162
883	161	934	163
884	161	935	163
885	161	936	163
886	161	937	163
887	161	938	163
888	161	939	163
889	161	940	163
890	161	941	163
891	161	942	163
892	162	943	163
893	162	944	163
894	162	945	163
895	162	946	163
896	162	947	163
897	162	948	163
898	162	949	163
899	162	950	163
900	162	951	163
901	162	952	163
902	162	953	163
903	162	954	163
904	162	955	163
905	162	956	163
906	162	957	163
907	162	958	163
908	162	959	163
909	162		
910	162		
911	162		
912	162		
913	162		
914	162		
915	162		
916	162		
917	162		
918	162		
919	162		
920	162		

Table E-1. Range Normalization (con't)

CELL #	CONTENTS	CELL #	CONTENTS
960	163	1010	164
961	163	1011	164
962	163	1012	164
963	163	1013	164
964	163	1014	164
965	163	1015	164
966	163	1016	164
967	163	1017	164
968	163	1018	164
969	163	1019	164
970	163	1020	164
971	163	1021	164
972	163	1022	164
973	163	1023	164
974	163		
975	163		
976	163		
977	163		
978	164		
979	164		
980	164		
981	164		
982	164		
983	164		
984	164		
985	164		
986	164		
987	164		
988	164		
989	164		
990	164		
991	164		
992	164		
993	164		
994	164		
995	164		
996	164		
997	164		
998	164		
999	164		
1000	164		
1001	164		
1002	164		
1003	164		
1004	164		
1005	164		
1006	164		
1007	164		
1008	164		
1009	164		

Table E-2. Range Normalization Tables ($T_p = 2\mu S$)

CELL #	CONTENTS	CELL #	CONTENTS	CELL #	CONTENTS
1	31	50	115	100	130
2	45	51	116	101	130
3	54	52	116	102	131
4	60	53	116	103	131
5	65	54	117	104	131
6	69	55	117	105	131
7	73	56	118	106	131
8	76	57	118	107	132
9	78	58	118	108	132
10	80	59	119	109	132
11	82	60	119	110	132
12	84	61	119	111	132
13	86	62	120	112	133
14	88	63	120	113	133
15	89	64	121	114	133
16	90	65	121	115	133
17	92	66	121	116	133
18	93	67	121	117	134
19	94	68	122	118	134
20	95	69	122	119	134
21	96	70	122	120	134
22	97	71	123	121	134
23	98	72	123	122	134
24	99	73	123	123	135
25	100	74	124	124	135
26	101	75	124	125	135
27	102	76	124	126	135
28	103	77	125	127	135
29	103	78	125	128	136
30	104	79	125	129	136
31	105	80	125	130	136
32	106	81	126	131	136
33	106	82	126	132	136
34	107	83	126	133	136
35	107	84	126	134	136
36	108	85	127	135	137
37	109	86	127	136	137
38	109	87	127	137	137
39	110	88	127	138	137
40	110	89	128	139	137
41	111	90	128	140	137
42	111	91	128	141	138
43	112	92	128	142	138
44	112	93	129	143	138
45	113	94	129	144	138
46	113	95	129	145	138
47	114	96	129	146	138
48	114	97	129	147	138
49	115	98	130	148	139
		99	130	149	139

Table E-2. Range Normalization Tables (con't)

CELL #	CONTENTS	CELL #	CONTENTS
150	139	200	145
151	139	201	145
152	139	202	145
153	139	203	145
154	140	204	146
155	140	205	146
156	140	206	146
157	140	207	146
158	140	208	146
159	140	209	146
160	140	210	146
161	140	211	146
162	141	212	146
163	141	213	147
164	141	214	147
165	141	215	147
166	141	216	147
167	141	217	147
168	141	218	147
169	142	219	147
170	142	220	147
171	142	221	147
172	142	222	147
173	142	223	148
174	142	224	148
175	142	225	148
176	142	226	148
177	143	227	148
178	143	228	148
179	143	229	148
180	143	230	148
181	143	231	148
182	143	232	148
183	143	233	148
184	143	234	149
185	143	235	149
186	144	236	149
187	144	237	149
188	144	238	149
189	144	239	149
190	144		
191	144		
192	144		
193	144		
194	144		
195	145		
196	145		
197	145		
198	145		
199	145		

Table E-2. Range Normalization Tables (con't)

CELL #	CONTENTS	CELL #	CONTENTS	CELL #	CONTENTS
240	149	290	153	340	157
241	149	291	153	341	157
242	149	292	153	342	157
243	149	293	153	343	157
244	149	294	153	344	157
245	150	295	154	345	157
246	150	296	154	346	157
247	150	297	154	347	157
248	150	298	154	348	157
249	150	299	154	349	157
250	150	300	154	350	157
251	150	301	154	351	157
252	150	302	154	352	157
253	150	303	154	353	157
254	150	304	154	354	158
255	150	305	154	355	158
256	150	306	154	356	158
257	151	307	154	357	158
258	151	308	155	358	158
259	151	309	155	359	158
260	151	310	155	360	158
261	151	311	155	361	158
262	151	312	155	362	158
263	151	313	155	363	158
264	151	314	155	364	158
265	151	315	155	365	158
266	151	316	155	366	158
267	151	317	155	367	158
268	151	318	155	368	158
269	152	319	155	369	158
270	152	320	155	370	158
271	152	321	155	371	159
272	152	322	155	372	159
273	152	323	156	373	159
274	152	324	156	374	159
275	152	325	156	375	159
276	152	326	156	376	159
277	152	327	156	377	159
278	152	328	156	378	159
279	152	329	156	379	159
280	152	330	156	380	159
281	153	331	156	381	159
282	153	332	156	382	159
283	153	333	156	383	159
284	153	334	156	384	159
285	153	335	156	385	159
286	153	336	156	386	159
287	153	337	156	387	159
288	153	338	157	388	159
289	153	339	157	389	160

Table E-2. Range Normalization Tables (con't)

CELL #	CONTENTS	CELL #	CONTENTS
390	160	440	162
391	160	441	162
392	160	442	162
393	160	443	162
394	160	444	162
395	160	445	162
396	160	446	163
397	160	447	163
398	160	448	163
399	160	449	163
400	160	450	163
401	160	451	163
402	160	452	163
403	160	453	163
404	160	454	163
405	160	455	163
406	160	456	163
407	161	457	163
408	161	458	163
409	161	459	163
410	161	460	163
411	161	461	163
412	161	462	163
413	161	463	163
414	161	464	163
415	161	465	163
416	161	466	163
417	161	467	164
418	161	468	164
419	161	469	164
420	161	470	164
421	161	471	164
422	161	472	164
423	161	473	164
424	161	474	164
425	161	475	164
426	162	476	164
427	162	477	164
428	162	478	164
429	162	479	164
430	162		
431	162		
432	162		
433	162		
434	162		
435	162		
436	162		
437	162		
438	162		
439	162		

Table E-2. Range Normalization Tables (con't)

CELL #	CONTENTS	CELL #	CONTENTS	CELL #	CONTENTS
480	164	530	166	580	168
481	164	531	166	581	168
482	164	532	166	582	168
483	164	533	166	583	168
484	164	534	166	584	168
485	164	535	166	585	168
486	164	536	166	586	168
487	164	537	167	587	168
488	164	538	167	588	168
489	165	539	167	589	169
490	165	540	167	590	169
491	165	541	167	591	169
492	165	542	167	592	169
493	165	543	167	593	169
494	165	544	167	594	169
495	165	545	167	595	169
496	165	546	167	596	169
497	165	547	167	597	169
498	165	548	167	598	169
499	165	549	167	599	169
500	165	550	167	600	169
501	165	551	167	601	169
502	165	552	167	602	169
503	165	553	167	603	169
504	165	554	167	604	169
505	165	555	167	605	169
506	165	556	167	606	169
507	165	557	167	607	169
508	165	558	167	608	169
509	165	559	167	609	169
510	165	560	167	610	169
511	165	561	167	611	169
512	166	562	168	612	169
513	166	563	168	613	169
514	166	564	168	614	169
515	166	565	168	615	169
516	166	566	168	616	170
517	166	567	168	617	170
518	166	568	168	618	170
519	166	569	168	619	170
520	166	570	168	620	170
521	166	571	168	621	170
522	166	572	168	622	170
523	166	573	168	623	170
524	166	574	168	624	170
525	166	575	168	625	170
526	166	576	168	626	170
527	166	577	168	627	170
528	166	578	168	628	170
529	166	579	168	629	170

Table E-2. Range Normalization Tables (con't)

CELL #	CONTENTS	CELL #	CONTENTS
630	170	680	172
631	170	681	172
632	170	682	172
633	170	683	172
634	170	684	172
635	170	685	172
636	170	686	172
637	170	687	172
638	170	688	172
639	170	689	172
640	170	690	172
641	170	691	172
642	170	692	172
643	170	693	172
644	170	694	172
645	170	695	172
646	171	696	172
647	171	697	172
648	171	698	172
649	171	699	172
650	171	700	172
651	171	701	172
652	171	702	172
653	171	703	172
654	171	704	172
655	171	705	172
656	171	706	172
657	171	707	172
658	171	708	173
659	171	709	173
660	171	710	173
661	171	711	173
662	171	712	173
663	171	713	173
664	171	714	173
665	171	715	173
666	171	716	173
667	171	717	173
668	171	718	173
669	171	719	173
670	171		
671	171		
672	171		
673	171		
674	171		
675	171		
676	172		
677	172		
678	172		
679	172		

Table E-2. Range Normalization Tables (con't)

CELL #	CONTENTS	CELL #	CONTENTS	CELL #	CONTENTS
720	173	770	174	820	176
721	173	771	174	821	176
722	173	772	174	822	176
723	173	773	174	823	176
724	173	774	174	824	176
725	173	775	174	825	176
726	173	776	174	826	176
727	173	777	175	827	176
728	173	778	175	828	176
729	173	779	175	829	176
730	173	780	175	830	176
731	173	781	175	831	176
732	173	782	175	832	176
733	173	783	175	833	176
734	173	784	175	834	176
735	173	785	175	835	176
736	173	786	175	836	176
737	173	787	175	837	176
738	173	788	175	838	176
739	173	789	175	839	176
740	173	790	175	840	176
741	173	791	175	841	176
742	174	792	175	842	176
743	174	793	175	843	176
744	174	794	175	844	176
745	174	795	175	845	176
746	174	796	175	846	176
747	174	797	175	847	176
748	174	798	175	848	176
749	174	799	175	849	176
750	174	800	175	850	176
751	174	801	175	851	176
752	174	802	175	852	177
753	174	803	175	853	177
754	174	804	175	854	177
755	174	805	175	855	177
756	174	806	175	856	177
757	174	807	175	857	177
758	174	808	175	858	177
759	174	809	175	859	177
760	174	810	175	860	177
761	174	811	175	861	177
762	174	812	175	862	177
763	174	813	176	863	177
764	174	814	176	864	177
765	174	815	176	865	177
766	174	816	176	866	177
767	174	817	176	867	177
768	174	818	176	868	177
769	174	819	176	869	177

Table E-2. Range Normalization Tables (con't)

CELL #	CONTENTS	CELL #	CONTENTS
870	177	920	178
871	177	921	178
872	177	922	178
873	177	923	178
874	177	924	178
875	177	925	178
876	177	926	178
877	177	927	178
878	177	928	178
879	177	929	178
880	177	930	178
881	177	931	178
882	177	932	178
883	177	933	178
884	177	934	179
885	177	935	179
886	177	936	179
887	177	937	179
888	177	938	179
889	177	939	179
890	177	940	179
891	177	941	179
892	178	942	179
893	178	943	179
894	178	944	179
895	178	945	179
896	178	946	179
897	178	947	179
898	178	948	179
899	178	949	179
900	178	950	179
901	178	951	179
902	178	952	179
903	178	953	179
904	178	954	179
905	178	955	179
906	178	956	179
907	178	957	179
908	178	958	179
909	178	959	179
910	178		
911	178		
912	178		
913	178		
914	178		
915	178		
916	178		
917	178		
918	178		
919	178		

Table E-2. Range Normalization Tables (con't)

CELL #	CONTENTS	CELL #	CONTENTS
960	179	1010	180
961	179	1011	180
962	179	1012	180
963	179	1013	180
964	179	1014	180
965	179	1015	180
966	179	1016	180
967	179	1017	180
968	179	1018	180
969	179	1019	180
970	179	1020	180
971	179	1021	180
972	179	1022	180
973	179	1023	180
974	179		
975	179		
976	179		
977	179		
978	180		
979	180		
980	180		
981	180		
982	180		
983	180		
984	180		
985	180		
986	180		
987	180		
988	180		
989	180		
990	180		
991	180		
992	180		
993	180		
994	180		
995	180		
996	180		
997	180		
998	180		
999	180		
1000	180		
1001	180		
1002	180		
1003	180		
1004	180		
1005	180		
1006	180		
1007	180		
1008	180		
1009	180		

APPENDIX F

Cables and Interconnect Specifications

Table F-1. Intecolor-TIE Interconnect

<u>Signal Name</u>	<u>Intecolor Port</u>	<u>Intecolor Connector (24-Bit I/O)</u>	<u>TIE Connector Pin#</u>	<u>TIE Pin#</u>
DATA0	A0	2	J3-2	AE52
DATA1	A1	14	J3-16	AF51
DATA2	A2	3	J3-3	AE53
DATA3	A3	15	J3-15	AF52
DATA4	A4	4	J3-4	AE54
DATA5	A5	16	J3-14	AF53
DATA6	A6	5	J3-5	AE55
DATA7	A7	17	J3-13	AF54
ITXRST	B0	10	J4-2	AG52
IRXRST	B1	9	J4-1	AG51
ISELBUS	B2	21	J3-9	AF58
ILRADR	B3	8	J3-8	AE58
ILIOADR	B4	20	J3-10	AF57
MSPDIS	B5	7	J3-7	AE57
(SPARE)	B6	19	J3-11	AF56
ITENBL	B7	6	J3-6	AE56
ITRBULD	C0	12	J4-4	AG54
ITXST	C1	23	J4-15	AH52
IRDIOBUF	C2	11	J4-3	AG53
IRIOSTAT	C3	22	J4-16	AH51
TXINT	C4	1	J3-1	AE51
RXINT	C5	13	J4-5	AG55
LSP/MSPC	C6	24	J4-14	AH53
(SPARE)	C7	25	J4-13	AH54
GND	GND	18	J3-12	AF55

Table F-2. TIE-CE Interconnect

<u>Signal Name</u>	<u>Pin# (Bus A -> J1) (Bus B -> J2)</u>	<u>Signal Name</u>	<u>Pin# (Bus A -> J1) (Bus B -> J2)</u>
(SPARE)	26	GND	01
(SPARE)	27	"	02
(SPARE)	28	"	03
OCCN	29	"	04
RDYN	30	"	05
OPARN	31	"	06
BUSYN	32	"	07
ACKN	33	"	08
DATA00N	34	"	09
DATA01N	35	"	10
DATA02N	36	"	11
DATA03N	37	"	12
DATA04N	38	"	13
DATA05N	39	"	14
DATA06N	40	"	15
DATA07N	41	"	16
DATA08N	42	"	17
DATA09N	43	"	18
DATA10N	44	"	19
DATA11N	45	"	20
DATA12N	46	"	21
DATA13N	47	"	22
DATA14N	48	"	23
DATA15N	49	"	24
(UNUSED)	50	GND	25

Table F-3. CE Bus Cable Definitions

<u>Signal Name</u>	<u>Pin# (Bus A -> J2) (Bus B -> J3)</u>	<u>Signal Name</u>	<u>Pin# (Bus A -> J2) (Bus B -> J3)</u>
(SPARE)	26	GND	01
(SPARE)	27	"	02
(SPARE)	28	"	03
OCCN	29	"	04
RDYN	30	"	05
OPARN	31	"	06
BUSYN	32	"	07
ACKN	33	"	08
DATA00N	34	"	09
DATA01N	35	"	10
DATA02N	36	"	11
DATA03N	37	"	12
DATA04N	38	"	13
DATA05N	39	"	14
DATA06N	40	"	15
DATA07N	41	"	16
DATA08N	42	"	17
DATA09N	43	"	18
DATA10N	44	"	19
DATA11N	45	"	20
DATA12N	46	"	21
DATA13N	47	"	22
DATA14N	48	"	23
DATA15N	49	"	24
(UNUSED)	50	GND	25

Table F-4. IOC-Input Synchronizer Interconnect

<u>Signal Name</u>	<u>IS Pin# (slot T13)</u>	<u>Signal Name</u>	<u>IS Pin# (slot T13)</u>
SEL0N	46	+5V	37
SEL1N	47	"	38
SEL2N	48	"	39
OCCN	49	"	40
RDYN	50	"	78
OPARN	51	"	79
BUSYN	--	"	80
ACKN	53	"	117
DATA00N	54	"	118
DATA01N	55	"	119
DATA02N	56	+5V	120
DATA03N	57	GND	1
DATA04N	58	"	2
DATA05N	59	"	3
DATA06N	60	"	4
DATA07N	61	"	41
DATA08N	62	"	42
DATA09N	63	"	43
DATA10N	64	"	81
DATA11N	65	"	82
DATA12N	66	"	83
DATA13N	67	GND	84
DATA14N	68	(SPARE)	5
DATA15N	69	"	6
SEL3N	--	"	7

Table F-4. IOC-Input Synchronizer Interconnect (con't)

<u>Signal Name</u>	<u>IS Pin# (slot T13)</u>	<u>Signal Name</u>	<u>IS Pin# (slot T13)</u>
GND	86	"	8
"	87	"	9
"	88	"	10
"	89	"	11
"	90	"	12
"	91	"	13
"	92	"	14
"	93	"	15
"	94	"	16
"	95	(SPARE)	44
"	96		
"	97		
"	98		
"	99		
"	100		
"	101		
"	102		
"	103		
"	104		
"	105		
"	106		
"	107		
"	108		
"	109		
GND	110		

Table F-5. Input Synchronizer - PPP Interconnect

<u>Signal Name</u>	<u>Pin (Slot T13)</u>	<u>Slot Number</u>	<u>Pin Number</u>
LP08	17	R4	18
LP07	18	R4	17
LP06	19	R4	16
LP05	20	R4	15
LP04	21	R4	14
LP03	22	R4	13
LP02	23	R4	12
LP01	24	R4	11
QC08	25	F9	18
QC07	26	F9	17
QC06	27	F9	16
QC05	28	F9	15
QC04	29	F9	14
QC03	30	F9	13
QC02	31	F9	12
QC01	32	F9	53
IC08	33	F10	18
IC07	34	F10	17
IC06	35	F10	16
IC05	36	F10	15
GND	37		
GND	38		
GND	39		
GND	40		
(UNUSED)	71		
NRC0	72	R7	7
IC01	73	F7	13
IC02	74	F10	12
IC03	75	F10	13
IC04	76	F10	14
PHI3SP	77	---(Not yet used)---	
(UNUSED)	78		
(UNUSED)	79		
(UNUSED)	80		
(UNUSED)	111		
NRC1	112	R7	9
TPO	113	R7	20
TP1	114	R7	19
PHI3SIQ	115	R7	116
RDRTRIG	116	R7	32
(UNUSED)	117		
(UNUSED)	118		
(UNUSED)	119		
(UNUSED)	120		

Table F-6. Output Synchronizer - PPP Interface Interconnect

<u>Output Synchronizer</u>		<u>PPP Interface</u>	
<u>Signal Name</u>	<u>Connector Pin</u>	<u>Signal Name</u>	<u>Connector Pin</u>
GND	AJ1-14	GND	J3-14
GND	AJ1-13	GND	J3-13
(SPARE)	AJ-12	---	J3-12
M0(Sign)	AJ1-11	MEANSIGN	J3-11
M10(LSB)	AJ1-10	MEANLSB	J3-10
M9	AJ1-9		J3-9
M8	AJ1-8		J3-8
M7	AJ1-7		J3-7
M6	AJ1-6		J3-6
M5	AJ1-5		J3-5
M4	AJ1-4		J3-4
M3	AJ1-3		J3-3
M2	AJ1-2		J3-2
M1(MSB)	AJ1-1	MEANMSB	J3-1
GND	AJ2-10	GND	J2-10
GND	AJ2-9	GND	J2-9
S8(LSB)	AJ2-8	VARLSB	J2-8
S7	AJ2-7		J2-7
S6	AJ2-6		J2-6
S5	AJ2-5		J2-5
S4	AJ2-4		J2-4
S3	AJ2-3		J2-3
S2	AJ2-2		J2-2
S1(Sign)	AJ2-1	VARMSB	J2-1

Table F-6. Output Synchronizer - PPP Interface Interconnect (Continued)

Output Synchronizer		PPP Interface	
Signal Name	Connector Pin	Signal Name	Connector Pin
(SPARE)	AJ2-13		J1-13
(SPARE)	AJ2-12		J1-12
(SPARE)	AJ2-11		J1-11
GND	AJ2-23	GND	J1-10
P9(LSB)	AJ2-22	POWERLSB	J1-9
P8	AJ2-21		J1-8
P7	AJ2-20		J1-7
P6	AJ2-19		J1-6
P5	AJ2-18		J1-5
P4	AJ2-17		J1-4
P3	AJ2-16		J1-3
P2	AJ2-15		J1-2
P1(MSB)	AJ2-14	POWERMSB	J1-1
(SPARE)	AJ2-26		J1-14
GND	BJ2-13	GND	J4-13
VCC	BJ2-12	VCC	J4-12
VCC	BJ2-11	VCC	J4-11
VCC	BJ2-10	VCC	J4-10
GND	BJ2-14	GND	J4-9
GND	BJ2-17	GND	J4-8
GND	BJ2-7	GND	J4-7
NRCB	BJ2-6	NRCB	J4-6
NRCA	BJ2-5	NRCA	J4-5
DUMP PULSE	BJ2-4	DUMP PULSE	J4-4
CLKWIDTHB	BJ2-3	CLKWIDTHB	J4-3
CLKWIDTHA	BJ2-2	CLKWIDTHA	J4-2
PH1L	BJ2-1	PH1L	J4-1
(SPARE)	BJ2-26		J4-14

Table F-7. IOC - OS Interconnect

<u>Signal Name</u>	<u>OS Pin Number</u>	<u>Signal Name</u>	<u>OS Pin Number</u>
SEL0N	CJ1-1	GND	CJ1-14
SEL1N	CJ1-2	"	CJ1-15
SEL2N	CJ1-3	"	CJ1-16
OCCN	CJ1-4	"	CJ1-17
RDYN	CJ1-5	"	CJ1-18
OPARN	CJ1-6	"	CJ1-19
BUSYN	CJ1-7	"	CJ1-20
ACKN	CJ1-8	"	CJ1-21
DATA00N	CJ1-9	"	CJ1-22
DATA01N	CJ1-10	"	CJ1-23
DATA02N	CJ1-11	"	CJ1-24
DATA03N	CJ1-12	"	CJ1-25
DATA04N	CJ1-13	"	CJ1-26
DATA05N	CJ2-1	"	CJ2-14
DATA06N	CJ2-2	"	CJ2-15
DATA07N	CJ2-3	"	CJ2-16
DATA08N	CJ2-4	"	CJ2-17
DATA09N	CJ2-5	"	CJ2-18
DATA10N	CJ2-6	"	CJ2-19
DATA11N	CJ2-7	"	CJ2-20
DATA12N	CJ2-8	"	CJ2-21
DATA13N	CJ2-9	"	CJ2-22
DATA14N	CJ2-10	"	CJ2-23
DATA15N	CJ2-11	"	CJ2-24
SEL3N	CJ2-12	GND	CJ2-25
(UNUSED)	CJ2-13	(UNUSED)	CJ2-26

APPENDIX G
FTWRP Configuration

Table G-1. List of Schematics

<u>Drawing No.</u>	<u>Title</u>	<u>Size</u>	<u>Sheets</u>
SD1062804	Input Synchronizer Schematic Diagram	E	2
LY977725-D	Common Element Layout	E	1
SD977725-D	Common Element Schematic	E	8
LY977728	Input/Output Controller Layout	E	1
SD977728	Input/Output Controller Schematic Diagram	E	6
LY1062802	Output Synchronizer Layout	D	1
SD1062802	Output Synchronizer Schematic Diagram	D	1

Table G-2. FTWRP Software Configuration - Source Tapes

<u>Tape No.</u>	<u>Description</u>	<u>Title</u>
STG123419	9-Track Labelled 1600 BPI Tape	FTWRP Applications
STG123421	9-Track Labelled 1600 BPI Tape	FTWRP Support Programs
STG123423	9-Track Labelled 1600 BPI Tape	Intecolor Programs
STG123426	9-Track Labelled 1600 BPI Tape	DOS-0 Source
STG123427	9-Track Labelled 1600 BPI Tape	FTWRP CE Microcode Source

Table G-3 FTWRP Software Configuration - Supporting Documents

<u>Drawing No.</u>	<u>Title</u>	<u>Size</u>
PSG123419	Program Summary for FTWRP Applications	A
SMG123419	Methods Sheet for FTWRP Applications	A
PSG123421	Program Summary for FTWRP Support Programs	A
SMG123421	Methods Sheet for FTWRP Support Programs	A
PSG123423	Program Summary for Intecolor Programs	A
SMG123423	Methods Sheet for Intecolor Programs	A
PSG123426	Program Summary for DOS-0	A
SMG123426	Methods Sheet for DOS-0	A
PSG123427	Program Summary for FTWRP Microcode	A
SMG123427	Methods Sheet for FTWRP Microcode	A
MLG123427	Maintenance Log for FTWRP Microcode	A

Table G-4 FTWRP Program Listings

<u>Listing No.</u>	<u>Size</u>	<u>Title</u>
PRG123426	B	DOS-0 Listing
PRG123427	B	FTWRP Microcode Listing
PRG123477	B	FTWRP Listing (Continuous Pulse Sequence)
PRG123478	B	Dual Listing (Dual Wavelength)
PRG123479	B	FTWRTTG Listing (Test-Data Generator)
PRG123480	B	FTWRTST Listing (Microcode Test Program)
PRG123481	B	IDOS-0 Listing
PRG123482	B	IDOS-1 Listing
PRG123483	B	Utility Listing

END

DATE
FILMED

1-82

DTIC